

AC

(19) 日本国特許庁 (J P)

(12) 公表特許公報 (A)

(11) 特許出願公表番号

特表平11-500551

(43) 公表日 平成11年(1999) 1月12日

(51) Int.Cl. ⁶	識別記号	F I
G 0 6 F 9/38	3 3 0	G 0 6 F 9/38
	3 8 0	3 3 0 K
		3 8 0 B
		3 8 0 C

審査請求 未請求 予備審査請求 有 (全 354 頁)

(21) 出願番号 特願平8-525085
 (86) (22) 出願日 平成8年(1996) 2月13日
 (85) 翻訳文提出日 平成9年(1997) 8月13日
 (86) 国際出願番号 P C T / U S 9 6 / 0 1 9 3 0
 (87) 国際公開番号 W O 9 6 / 2 5 7 0 5
 (87) 国際公開日 平成8年(1996) 8月22日
 (31) 優先権主張番号 0 8 / 3 9 0 , 8 8 5
 (32) 優先日 1995年2月14日
 (33) 優先権主張国 米国 (U S)
 (31) 優先権主張番号 0 8 / 3 9 8 , 2 9 9
 (32) 優先日 1995年3月3日
 (33) 優先権主張国 米国 (U S)

(71) 出願人 富士通株式会社
 神奈川県川崎市中原区上小田中4丁目1番
 1号
 (72) 発明者 シェン, ジーン ダブリュ.
 アメリカ合衆国, カリフォルニア 94043,
 マウンテン ビュー, セントラル アベニ
 ュ 181エー
 (72) 発明者 スゼエトー, ジョン
 アメリカ合衆国, カリフォルニア 94610,
 オークランド, イースト サーティーフォー
 ース ストリート 1217
 (74) 代理人 弁理士 石田 敬 (外3名)

最終頁に続く

(54) 【発明の名称】 特殊機能を提供する高性能投機的実行プロセッサの構造及び方法

(57) 【要約】

数多くの特殊なプロセッサ機能及び能力を提供するための構造及び方法を内含する、中央処理ユニット(CPU)といった高性能プロセッサが開示されている。これらの構造及び方法には、制限的な意味はないものの、(1) 精確な状態を維持しながらロード/ストア命令を含む長待ち時間命令を攻撃的にスケジュールし；(2) 任意の命令境界において精確な状態を維持し回復し；(3) 精確な状態を維持するべく命令状態をトラッキングし；(4) 精確な状態を維持するべく命令をチェックポイントニングし；(5) タイムアウトチェックポイントを新規作成し維持し使用し；(6) 浮動小数点例外をトラッキングし；(7) リネーム可能なトラップスタックを新規作成し維持し使用し；(8) 複数の同時未解決分岐評価のためのウォッチポイントを新規作成し維持し使用し；(9) 精確な状態を維持するために命令状態をトラッキングし；(10) 精確な状態を維持する一方でプロセッサのスループットを増大させるための構造及び方法が含まれている。

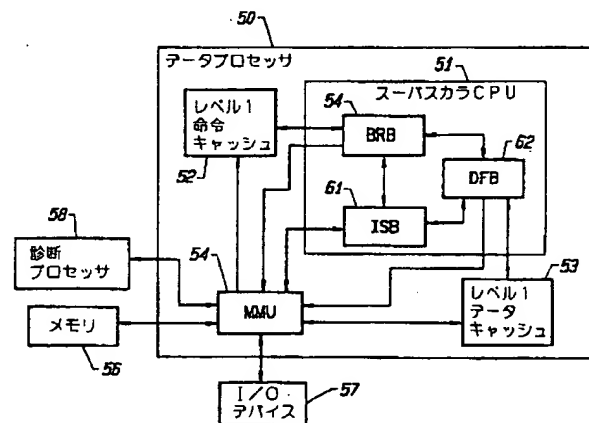


FIG. 4

【特許請求の範囲】

1. データ記憶装置、命令発行ユニット、命令実行ユニット及び命令発行・実行スケジューラを有する、命令発行ユニットが発行した命令を実行するための中央処理ユニットの中で、プロセッサ内の投機的命令実行をトラッキングするための方法において、

- ー データ記憶装置内のデータ構造を規定する段階；
 - ー 発行ユニットによって発行された各命令に対して識別タグを割当てする段階；
 - ー 割当てられたタグに基づいて各々の発行済み命令に対して、データ構造内に記憶された1つの活動ビットを結びつける段階；
 - ー 命令が発行された時点で、データ構造内に活動ビットをセットする段階；及び
 - ー 命令がエラー無しで実行を完了した時点でデータ構造内の活動ビットをクリアする段階、
- を含んで成る方法。

2. データ構造には、 n 個のアドレス可能な場所 $0, 1, 2, \dots, n-2, n-1$ をもつ円形データ構造が含まれており、ここでアドレス可能な場所 $n-1$ は、論理的にアドレス可能な場所 0 に隣接しており、データ構造内の一つの活動ビットを各々の発行済み命令と結びつける段階が、 n 個の場所のうちの固有の1つの場所の中の活動ビットを各々の発行済み命令と結びつけることを含んで成る請求項1に記載の方法。

3. 識別タグが数値的シリアル番号であり各々の発行済み命令に対して1つの識別タグを割当てする段階には、各々の命令が発行されるにつれてその後発行された命令に対し単調に増大するシリアル番号（モジュロ n ）を割当てることが含まれている請求項2に記載の

方法。

4. ー 付随する活動ビットの状態に基づいて各々の発行済み命令の実行状況をトラッキングする段階；
- ー 活動中の命令に付随する最小のシリアル番号を識別するべく単調にシリアル

番号が増大する順で円形データ構造内の各々の活動ビットを評価し、かくして最も早期に発行されなお活動中の命令を決定する段階；及び

- － 取消しが不可能であるように、前記最小の活動中のシリアル番号に付随するシリアル番号よりも小さいシリアル番号をもつ命令を完遂する段階；及び
- － 識別された最小のシリアル番号に基づいて完遂済み命令に割振られたプロセッサ資源を再生する段階；

をさらに含んで成る請求項3に記載の方法。

5. 再生された資源には、退去済み命令に付随するデータ構造の中の記憶場所及び命令シリアル番号が含まれる請求項4に記載の方法。

6. 投機的out-of-order実行プロセッサの中で、命令状況をトラッキングするための方法において、

- － 各々の発行済み命令について活動状況データを記憶するための複数の記憶場所をもつプロセッサ内のアドレス可能なデータ構造を提供する段階；
- － 各々の発行済み命令に対し1つの識別タグを割当てする段階；
- － 複数の記憶場所のうちの1つと、各々の割当てられたタグを結びつける段階；
- － 特定の命令が発行された時点で活動中であるものとして特定の発行済み命令を識別するデータを記憶する段階；
- － 特定の命令の完了を検出する段階；及び

－ 特定の命令が、実行条件の予め定められたセットのうちのいずれか1つが発生することなく実行を完了した時点で、特定の命令を識別するデータを記憶する段階、

を含んで成る方法。

7. 予め定められた実行条件セットには、命令実行エラー、例外、ハードウェア故障、分岐命令誤予測が含まれ、エラー条件が検出される請求項6に記載の方法。

8. 命令発行スケジューラ及び命令発行ユニットを有する投機的out-of-orderプロセッサの中で、発行済み命令の状況をトラッキングするための方法において

- 、
- 命令活動状況データを記憶するための n - 場所データ構造を提供する段階；
- n - 場所データ構造内の n 個の場所の各々について固有の記憶場所アドレスを規定する段階；
- 発行ユニットによって発行された各々の命令を、固有の記憶場所アドレスのうちの 1 つと結びつける段階；
- 命令が活動中であることを表示するべく命令に付随するデータ構造の場所においてデータ構造内に活動状況データを記憶する段階；
- 各々の発行済み命令について、命令実行完了及びそれに付随するあらゆるエラー状況を検出する段階；
- 命令がエラー無しで完了した場合、命令が現在非活動中であることを表示するための付随する記憶場所の中の活動状況データを更新し、命令がエラーを伴って完了した場合、命令がなおも活動中であることを表示するため付随する記憶場所の中にもとの活動状況データを保持する段階；及び
- 命令発行スケジューラが、当時非活動中であった以前に発行さ

れた命令の状況とは無関係にさらなる命令を実行のためのスケジュールできるような形で、命令発行スケジューラに対し命令活動状況データを伝送する段階、を含んで成る方法。

9. — プロセッサ内のエラーハンドラユニットに対し命令のうちの任意のものの実行完了に付随するエラー状況を伝達する段階、及び

- 伝達されたエラー状況に応じて命令発行及び実行シーケンスのスケジューリングを修正する段階、

を含んで成る請求項 8 に記載の方法。

10. 命令を発行するための命令発行ユニット及びプロセッサ内にデータを記憶するためのデータ記憶装置をもつ投機的out-of-order実行プロセッサの中で、精確なプロセッサ状態をトラッキングするための方法において、

- プロセッサのデータ記憶装置内に n 個のアドレス可能なデータ記憶場所をもつ第 1 のデータ構造を規定する段階；

- ー 発行済み命令に対しひきつづき割当てするべく複数の固有識別タグと割振る段階；
- ー 命令が発行された時点で、各々の命令に対して固有識別タグのうちの1つを割当てする段階；
- ー プロセッサ内の第1のデータ構造内のアドレス可能な記憶場所の1つと各々のタグを結びつける段階；
- ー 各々の命令について、各命令に対する命令活動状況の変化に応じて命令に付随する記憶場所の中に記憶されたデータを更新する段階；
- ー 記憶場所に複数のポインタを維持し、命令活動状況の変化に応じて、ポインタを移動させる段階；

を含んで成る方法。

11. 複数のポインタを維持する段階には、

- ー 複数の発行済み命令の各々について命令状況を決定するべくアドレス可能な記憶場所の中に記憶されたデータを評価する段階；
- ー 予め定められた実行状況のセットから選択された特定の実行状況を達成したその付随する識別タグを伴う命令を各々識別する複数の異なるプロセッサ条件インジケータを、アドレス可能な記憶場所の中に記憶されたデータに基づいて計算する段階；及び
- ー 前記プロセッサ内の第3のデータ記憶装置内にポインタとしてプロセッサ条件インジケータを記憶する段階；

を含んで成り、命令活動状況変化に応じてポインタを移動させる段階には、前記プロセッサ条件インジケータを更新するべく予め定められた時間的間隔で前記評価、異なるプロセッサ条件インジケータの計算及び記憶の段階を反復する段階が含まれている、請求項10に記載の方法。

12. 予め定められた実行状況セットには最後の発行済み命令、最後の完遂済み命令及び最後の退去済み命令が含まれる請求項10に記載の方法。

13. 予め定められた実行状況セットがさらに、最後の非メモリ参照命令及び最後の予測された分岐命令を含んでいる請求項10に記載の方法。

14. 前記状況データは、命令が発行された時点でセットされ、実行がエラー無く完了した時点でクリアされる活動中のビットを含んでいる請求項10に記載の方法。

15. 前記複数のポインタには、

- ー 最後の発行済み命令をポインタする第1のポインタ、及び
- ー 順序的に前に発行された全ての命令がエラー無しで完了してお

りそれ自体エラー無しで完了した最後の命令である最後の完遂済み命令をポインタする第2のポインタ、

が含まれている請求項14に記載の方法。

16. 前記複数のポインタにはさらに、

- ー 割振られたプロセッサ資源が再生された最後の命令である最後の再生済み命令をポイントする第3のポインタ

が含まれている請求項15に記載の方法。

17. 前記各々の識別タグが、単調に増加する順序の数値的シリアル番号のうちの1つであり、前記第1のデータ構造の中の前記アドレス可能な記憶場所が前記シリアル番号によってアドレスされる請求項16に記載の方法。

18. n 個のアドレス可能なデータ記憶場所をもつ前記第1のデータ構造が円形データ構造であり、より高い値のシリアル番号に対する前記ポインタの前進は、アドレス n からポインタを増分させることでこのポインタがアドレス0に置かれるような形でモジュロ n 算術演算で達成される請求項17に記載の方法。

19. 命令活動状況変化に応じて前記ポインタを移動させることによりこのポインタを維持する前記段階には、

- ー 発行された全ての命令について、1つのシリアル番号だけより高い命令シリアル番号（モジュロ n ）に向かって順方向に前記第1の発行済み命令ポインタを前進させる段階、
- ー 各記憶場所についての活動中のビットの状態のシリアル番号逐次評価及び第1の予め定められた規則に基づいてより高い命令シリアル番号（モジュロ n ）に向かって順方向に前記第2の完遂済み命令ポインタを前進させるものの、前記第

1のポイントをを超えて前記第2のポイントを前進させることはしない段階；及び

- ー 各場所についての活動中のビットの状態の通信番号逐次評価及

び第2の予め定められた規則に基づいてより高い命令シリアル番号（モジュロ n ）に向かって順方向に前記第3の退去済みのポイントを前進させるものの、前記第2のポイントをを超えて前記第3のポイントを前進させることはしない段階、が含まれている請求項15に記載の方法。

20. 前記第1の予め定められた規則には、

- ー マシクロックサイクルの各々全ての予め定められた数において、より低いアドレス場所にある全ての活動状態ビットが非活動「0」である最高のデータ記憶場所アドレスより低いものの、そのマシンサイクルでの前記第1のポイントのアドレスほどは高くない新しいアドレス場所まで順方向に前記第2のポイントを前進させる段階；

が含まれている請求項19に記載の方法。

21. 前記第2の予め定められた規則には、マシクロックサイクルの各々全ての予め定められた数において、より低いアドレス場所にある全ての活動状況ビットが非活動「0」である最高のデータ記憶場所アドレスよりも低いものの、そのマシンサイクルでの前記第2のポイントのアドレスほどは高くない新しいアドレス場所まで順方向に前記第3のポイントを前進させる段階が含まれている請求項19に記載の方法。

22. 前記ポイントが、命令発行、完遂及び退去の結果として移動させられる請求項19に記載の方法。

23. 前記第2及び第3のポイントは、例外、分岐命令該予測及びエラー条件が検出された命令を通過して移動させられない請求項19に記載の方法。

24. データ記憶手段及び命令発行、ディスパッチ、実行及び退出をスケジュールするための手段を有する投機的out-of-order実行プ

ロセッサの中で、命令の状況をトラッキングする方法において、

- ー 前記プロセッサ内で競合して目立つ命令の最大数を規定する n 個の固有命令

シリアル番号を割振る段階；

ー 前記プロセッサ内で前記データ記憶手段内に n 個の独立してアドレス可能な記憶場所をもつデータ構造を規定する段階；

ー 前記割振られたシリアル番号各々を前記アドレス可能な記憶場所に結びつける段階；及び

ー 前記プロセッサが発行する命令に応じて、前記シリアル番号の1つを識別タグとして各々の前記発行済み命令に割当て、この命令が実行を完了し退去させられて暫定的に固有の対応が各々の発行済み命令と前記記憶場所のうち固有のもの間に打ち立てられるようになるまで前記割当てを維持する段階、
を含んで成る方法。

25. ー 前記命令が発行された時点で前記命令に付随する前記データ記憶場所の中に活動中の状況インジケータを記憶する段階；

ー 前記命令がエラー無しで完了した時点で前記命令に付随する前記データ記憶場所内に非活動中の状況インジケータを記憶する段階；

ー 前記アドレス可能な記憶場所をポイントする複数のマシンポイントを規定する段階；及び

ー 命令状況及び予め定められた規則に応じて、前記複数のマシンポイントを移動させることにより各命令に割振られた命令状況及びマシン資源をトラッキングする段階；

をさらに含んで成る請求項24に記載の方法。

26. 前記データ記憶場所がマルチビットレジスタ内のビットであり、前記活動中の状況インジケータが1ビットであり、前記非活動中の状況インジケータが0ビットである請求項25に記載の方法

。

27. 前記複数のマシンポイントには、

ー 最後の発行済み命令をポイントする第1のポイント、

ー 順序的に早期の全てのin-order発行済み命令がエラー無しで完了していてそれ自体エラー無しで完了した最後の命令である最後の完遂済み命令をポイントす

る第2の最後の完遂済み命令ポインタ；及び

ー 命令に対してプロセッサが割振った全てのマシン資源及び順序的に早期の全てのin-order命令が再生された最後の命令をポイントする第3の退去及び再生ポインタ、

が含まれている請求項25に記載の方法。

28. 前記データ構造は、アドレス可能な場所 $n-1$ が論理的に場所0に隣接しているような形で n 個のアドレス可能な場所 $0, 1, 2, \dots, n-2, n-1$ をもつ円形データ構造であり、前記シリアル番号は連続する整数であり、前記シリアル番号を前記アドレス可能な記憶場所の1つと結びつける段階には、前記円形レジスタ内の第1のビット位置と最低のシリアル番号を結びつけること及び前記円形レジスタ内の n 番目のビット位置と前記最高のシリアル番号を結びつけることが含まれている、請求項27に記載の方法。

29. 前記複数のマシンポインタを移動させることによって各命令に割振られた命令状況及びマシン資源をトラッキングする前記段階には、

ー 第1の命令を発行する前に同じ初期記憶場所アドレスに前記第1、第2及び第3のポインタを初期設定する段階；

ー 発行された各々全ての命令について1つのシリアル番号だけ、より高い命令シリアル番号（モジュロ n ）に向かって順方向に前記第1の発行命令ポインタを前進させる段階；

ー 各記憶場所についての活動中のビットの状態のシリアル番号逐次評価及び第1の予め定められた規則に基づいてより高い命令シリアル番号（モジュロ n ）に向かって順方向に前記第2の完遂済み命令ポインタを前進させるものの、前記第1のポインタを超えて前記第2のポインタを前進させることはしない段階；及び

ー 各場所についての活動中のビットの状態の通信番号逐次評価及び第2の予め定められた規則に基づいてより高い命令シリアル番号（モジュロ n ）に向かって順方向に前記第3の退去済みのポインタを前進させるものの、前記第2のポインタを超えて前記第3のポインタを前進させることはしない段階、

が含まれ、前記ポインタをより高い値のシリアル番号へと前進させる段階は、ア

ドレス $n-1$ からポインタを増分させることによってこのポインタがアドレス n に置かれ、このポインタをアドレス n から増分させるとこのポインタがアドレス 0 に置かれるような形で、モジュロ n 算術演算を用いて前記円形データ構造中で達成される請求項28に記載の方法。

30. 前記第1の予め定められた規則には、

ー マシクロックサイクルの各々全ての予め定められた数において、より低いアドレス場所にある全ての活動状況ビットが非活動「0」である最高のデータ記憶場所アドレスより低いものの、そのマシンサイクルでの前記第1のポインタのアドレスほどは高くない新しいアドレス場所まで順方向に前記第2のポインタを前記させる段階；

が含まれ、ここで前記第2の予め定められた規則には、

ー マシクロックサイクルの各々全ての予め定められた数において、より低いアドレス場所にある全ての活動状況ビットが非活動状態にある最高のデータ記憶場所アドレスより低いものの、そのマシ

ンサイクルでの前記第2のポインタのアドレスほどは高くない新しいアドレス場所まで順方向に前記第3のポインタを前進させる段階；

が含まれている、請求項29に記載の方法。

31. 前記新しいアドレス場所は、前記状況活動ビットのブール演算に基づいて決定される請求項30に記載の方法。

32. 前記第1の予め定められた規則には、前記第2のポインタが前記予め定められた数のマシンサイクルの間に前進され得るアドレス数を、第1のアドレス最大数に制限する段階がさらに含まれており；前記第2の予め定められた規則には、前記第3のポインタが前記予め定められた数のマシンサイクルの間に前進され得るアドレス数を第2のアドレス最大数に制限する段階がさらに含まれており、ここで前記第2のアドレス最大数は前記第1のアドレス最大数より小さい、請求項30に記載の方法。

33. 前記第2及び第3のポインタは、例外、分岐命令誤予測及びエラー条件が検出された命令を通過して移動させられない、請求項30に記載の方法。

34. データ記憶手段、少なくとも1つの命令実行ユニット及び命令発行ユニットをもつプロセッサの中で、精確な状態を維持するための装置において、

- ー 各々の発行済み命令について前記実行ユニットから命令実行状況を受領するための手段；
- ー 前記プロセッサ内の命令の発行、実行、完了及び退去に基づいてマシン資源利用可能性情報をトラッキングし前記情報を前記命令発行ユニットに伝達するための手段；及び
- ー マシン資源がさらなる命令を処理するのに利用可能である場合に命令の発行を続行し、マシン資源が命令の処理に利用可能でない

場合にはさらなる命令の発行を停止させるべく、前記資源利用可能性情報に対し応答性をもつ前記命令発行ユニット内の手段、
を含んで成る装置。

35. マシン資源利用可能性情報をトラッキングするための前記手段には、

- ー 各々の発行済み命令について発行、実行、完了及び退去の状況情報を記憶するための前記データ記憶手段内に構成されたデータ構造；
 - ー 最後の発行済み命令を決定するため、順序的により早期の全ての命令がエラー無く完了しておりかつそれ自体エラー無く完了した最後の命令である最後の完遂済み命令を決定するため、そして割振りされたプロセッサ資源が再生された最後の命令である最後の再生済み命令を決定するための論理手段
- が含まれている請求項34に記載の装置。

36. 前記発行ユニットによって発行された予測された命令を識別するための手段；

- ー 予測を誤った実行済みの予測分岐命令を検出し、前記誤予測分岐からの回復を開始するための手段
 - ー 命令実行例外をとり扱うための手段、
- をさらに含んで成る請求項35に記載の装置。

37. データ記憶装置、命令発行ユニット、命令実行ユニット及び命令発行・実行スケジューラをもつ中央処理ユニットの中で、このプロセッサ内での投機的命

令実行をトラッキングするための装置において、

- ー 前記データ記憶装置内に構成されたデータ構造；
 - ー 命令発行オペレーションに応答して発行ユニットにより発行された各々の命令に対し識別タグを割当ててするための手段；
 - ー 割当てられたタグに基づいて各々の前記発行済み命令と付随するデータ構造内で記憶された活動ビットを結びつけるための手段；
 - ー 命令が発行された時点でデータ構造内の活動ビットをセットするための手段；
 - ー 命令がエラー無く実行を完了した時点でデータ構造内の活動ビットをクリアするための手段、
- を含んで成る装置。

38. 内部データ記憶装置、命令復号ユニットをもち外部メモリと連絡している投機的out-of-order実行プロセッサの中で、精確な例外モデルを維持するロード及びストア命令を含めたメモリ参照命令をトラッキングし攻撃的にスケジュールするための方法において、

- ー 前記プロセッサによる実行のための複数の命令を発行する段階；
- ー 前記発行済みの複数の命令のうちのいずれかが投機的に発行された命令であるかを識別する段階；
- ー 前記内部データ記憶装置内に、前記識別された投機的に発行された命令の各々に付随する投機的実行インジケータを記憶する段階；
- ー 前記発行済み命令のいずれかが外部メモリを参照するかを決定する段階；
- ー 前記内部データ記憶装置の中に前記決定されたメモリ参照命令に付随するメモリ参照命令インジケータを記憶する段階；
- ー 前記命令が発行された後前記複数の命令のうちの各々の命令の実行活動状況を監視する段階；
- ー 各々の発行済み命令について実行中に何らかのエラー条件が発生したか否かを確認し、実行中にエラーを経験した各々の命令のためにエラー状況を表示するエラー条件インジケータを生成する段階

；

- ー 前記発行済み命令の実行状況をトラッキングする段階；及び
 - ー その他の発行されたものの実行されていない命令の実行状況に基づいて、順序的により早期の発行済みの非メモリ参照命令に先立ち、out-of-order実行のために前記決定されたメモリ参照命令のうちの特定の 1 つの命令をスケジュールする段階であって、この実行状況には、投機的に発行された命令であるものとしての非メモリ参照命令の識別及び予め定められた実行完了状況を有するものとしての非メモリ参照命令の識別が含まれている、スケジュール段階、
- を含んで成る方法。

39. 前記実行状況をトラッキングする段階には；

- ー 命令シリアル番号により各命令を識別する段階；
- ー 最後の発行済み命令のシリアル番号に対応するポインタ値を前記レジスタのうちの第 1 のレジスタの中に記憶する段階；
- ー 最後の非活動化済み命令のシリアル番号に対応するポインタ値を前記レジスタのうちの第 2 のレジスタの中に記憶する段階であって、ここで非活動化済み命令は実行を完了したもののその結果が状態ライトバックされていない 1 つの命令であるような、段階；
- ー 最後の完遂済み命令のシリアル番号に対応するポインタ値を前記レジスタのうちの第 3 のレジスタ内に記憶する段階；
- ー 最後の退去済み命令のシリアル番号に対応するポインタ値を（前記レジスタのうちの第 1 のレジスタ内の記憶のシリアル番号に対応するポインタ値を）前記レジスタのうちの第 4 のレジスタ内に記憶する段階；
- ー 最後の予測された分岐命令のシリアル番号に対応するポインタ値を前記レジスタのうちの第 5 のレジスタ内に記憶する段階；
- ー 最後の非メモリ参照命令のシリアル番号に対応するポインタ値

を前記レジスタのうちの第 6 のレジスタ内に記憶する段階；及び

- ー 前記第 1、第 2、第 3、第 4、第 5 及び第 6 のポインタ値を評価することにより前記プロセッサの状況を確認する段階、

が含まれる請求項38に記載の方法。

40. 内部のレジスタと外部のメモリを含むデータ記憶手段、命令発行ユニット及び命令実行ユニットを有する投機的out-of-order実行プロセッサの中で、このプロセッサ内の精確なアーキテクチャ状態を維持しながら、順序的にout-of-orderで低待ち時間命令に先立って、アーキテクチャ状態を修正できる長待ち時間命令を攻撃的にスケジュールするための方法において、

- エラー無く実行を完了し取り消す必要が全く無い順序的に最後の連続的命令を識別する段階；
 - 実行の予測が誤った場合実行が取消されなくてはならない可能性のある、順序的に最も早期の投機的発行済みで未解決の予測された制御転送命令を識別する段階；
 - エラー無しで実行を完了し取消す必要の全く無い前記識別された順序的に最後の連続的命令と、実行の予測を誤った場合取消さなければならない可能性のある前記識別された順序的に最も早期の投機的発行済みで未解決の予測された制御転送命令の間に順序づけされたあらゆる長待ち時間命令を識別するための段階であって、この長待ち時間命令は、長待ち時間命令であるものとして予め指定された予め定められた命令セットと、発行済み命令を比較することによって識別される、識別段階；
 - 中間実行のため前記識別された長待ち時間命令をスケジュールする段階；
- を含んで成り、

かくして、誤予測をした可能性のある投機的発行済みの介入する

制御転送命令及びそうでなければ実行例外を生成した可能性のある命令とは無関係に、実行のためスケジュールされ得る長待ち時間命令のみを実行するためにスケジュールすることによって、長待ち時間命令が攻撃的に発行され精確な状態が維持されることになる方法。

41. エラー無しで実行を完了し取消しの必要が全く無い順序的に最後の連続的命令を識別する段階に先立って、

- n のアドレス可能場所 $0, 1, 2, \dots, n-2, n-1$ を有し、アドレス可

能場所 $n - 1$ がこのデータ記憶手段内で論理的にアドレス可能な場所 0 と隣接している、円形モジュロ n データ構造を規定する段階；

— 各命令が発行されるにつれて、各々のひきつづき発行された命令に対して単調に増大する（モジュロ n ）数値的シリアル番号識別タグを割当てする段階；

— 前記データ構造内の活動状況インジケータを前記タグに基づいて各々の発行済み命令と結びつける段階；

— 全て命令について、その命令が発行された時点ですでに発行されていたことを表示するため、各々の発行済み命令を用いて前記データ構造内の前記 n 個の場所のうちの固有の 1 つの場所で前記活動状況インジケータの最初の要素をセットする段階；

— 長待ち時間命令について、前記命令がその発行時点で長待ち時間命令であることを表示するべく、前記活動状況インジケータの第 2 の要素をセットする段階；

— 予測された結果に基づいて投機的に発行された命令について、その命令がその発行時点で投機的に発行された命令であることを表示するべく、前記活動状況インジケータの 1 要素をセットする段階；

— 前記命令がエラー無く完了した時点で、それがすでにエラー無く完了していたことを表示するべく、前記データ構造内の前記活動状況インジケータをクリアする段階；

— エラー無しで完了していた前記順序的に最後の連続的命令、前記順序的に最も早期の投機的未解決予測命令及び、前記識別された最後の連続的エラー無し完了済み命令と前記識別された早期未解決予測命令の間に順序づけされた前記長待ち時間命令を含む、発行済みの各々の命令についての実行状況を決定するべく、発行済みの複数の命令についての前記活動状況インジケータを評価することにより、発行済み命令の実行状況をトラッキングする段階、

をさらに含んで成る請求項 40 に記載の方法。

42. エラー無しで完了し取消しの必要が全く無く順序的に最後の連続的命令を識別するための前記段階には、前記データ構造内の 1 つのエントリをポイントし

、最後の非活動化済み命令を表示する第2のポインタを設定する段階が含まれており、前記最後の非活動化済み命令は、順序的に先行する全てのin-order命令もエラー無しで完了している。エラー無しで完了した順序的に最後のin-order命令であり、前記第2のポインタにより識別された命令よりも早い順序のin-order命令が、分岐誤予測及び例外とは無関係に実行可能であり；

— その実行を取消さなければならない可能性のある順序的に最も早期の投機的で未解決の予測された制御転送命令を識別する前記段階には、前記データ構造内の1つのエントリをポイントし、順序的に最も早期のin-order未解決分岐命令を表示する第1のポインタ (PBSN) を設定する段階が含まれており、前記第1のポインタによって識別された命令より順序的に早期のin-order命令が分岐誤予測とは無関係に実行可能であり；

— 前記識別された最後の連続的エラー無し完了命令と前記識別された最早期未解決予測制御転送命令の間に順序づけされたあらゆる長待ち時間命令を識別する前記段階が、前記第2のポインタ (CSN) と前記第3のポインタ (NMCSN) の間のあらゆる長待ち時間命令を実行のためにスケジュールすべく利用可能なものとして識別するが、前記第2のポインタ (NMCSN) と前記第1のポインタ (PBSN) の間のあらゆる非長待ち時間命令を実行のためにスケジュールすべく利用可能なものとしては識別しない段階を含んでいる、請求項41に記載の方法。

43. 前記識別段階は、各々のマシンサイクル毎に前記状況インジケータ要素を論理的に比較するブール論理比較演算を実行し、前記より高い数値的シリアル番号識別タグに対応する記憶場所に向かって前記ポインタの各々を前進させることによって実行される請求項42に記載の方法。

44. 前記長待ち時間命令が、前記外部メモリを参照する命令を含んでいる請求項40に記載の方法。

45. 前記外部メモリ参照命令には、ロード命令とストア命令が含まれている請求項44に記載の方法。

46. 内部レジスタ及び外部メモリを有する投機的out-of-order実行プロセッサの中で、プロセッサ内の精確な状態を危険にさらすことなく前記内部レジスタの

みを参照する命令に先立ちアーキテクチャ状態を修正できるメモリ参照命令をスケジュールするための方法において、

- ー 複数の命令についての状況情報を記憶するため、前記プロセッサ内のレジスタの中にデータ構造を設定する段階；
- ー 前記データ構造の中の1つのエントリをポイントし最早期未解決分岐命令を表示する第1の予測された分岐命令シリアル番号ポイ

ンタ (PBSN) を設定する段階であって、この第1のポインタによって識別された命令よりも順序的に早期のin-order命令は、分岐誤予測とは無関係に実行可能である、段階；

- ー 前記データ構造内の1つのエントリをポイントし、最後の非活動化された命令を表示する第2のポインタ (CSN) を設定する段階であって、この最後の非活動化された命令は、順序的に前のin-order命令が全てエラー無しで完了しておりそれ自体エラー無しで完了した順序的に最後のin-order命令であり、前記第2のポインタにより識別された命令より順序的に早期のin-order命令は、分岐誤予測及び例外とは無関係に実行可能である、段階；

- ー 前記データ構造内の1つのエントリをポイントし、メモリ参照命令以外の最後に完遂された命令を表示する第3の非メモリ完遂シリアル番号ポインタ (NMCSN) を設定し、前記第2のポインタ (CSN) を通過しさらに前記メモリ参照命令のうちの任意のものをあたかもそれらも同様に完遂されたかのごとくに通過して前進する段階；

- ー 前記第2のポインタ (CSN) と前記第3のポインタ (NMCSN) の間のあらゆるメモリ参照命令を実行のためにスケジュールすべく利用可能なものとして識別するものの、前記第2のポインタ (NMCSN) と前記第1のポインタ (PBSN) の間の任意のメモリ参照命令を実行のためスケジュールするのに利用可能なものとして識別しない段階；

を含んで成り、かくして

長待ち時間メモリ参照命令は短待ち時間レジスタ参照命令から結合解除され、

- ー 分岐誤予測及び実行例外とは無関係に実行のためにスケジュールされ得るメ

メモリ参照命令のみをスケジュールすることによって正確な状態が維持される、スケジュールリング方法。

47. 前記データ構造内の1つのエントリをポイントしメモリを参照する命令以外の最後の完遂済み命令を表示する前記第3のポインタ (NMCSN) を設定し、前記第2のポインタ (DSN) を通過し前記メモリを参照指示する命令をあたかもそれらが同様に完遂されたかのごとく通過して前進し、さらに、活動中の又は非活動化された状況インジケータのいずれを有するかとは無関係に非障害発生命令であるものと予め決定されたあらゆる命令を通過して前進する段階をさらに含んで成る請求項46に記載の方法。

48. 内部のレジスタと外部のメモリ、命令発行ユニット及び命令実行ユニットを有する投機的out-of-order実行プロセッサの中で、このプロセッサ内の正確な状態を危険にさらすことなくレジスタのみを参照する命令に先立つアーキテクチャ状態を修正することのできるメモリ参照命令を攻撃的にスケジュールする方法において；

- ー メモリ参照命令及び非メモリ参照命令上に第1の状況情報を記憶するための複数のアドレス可能なデータ記憶要素を含む第1のデータ構造を規定する段階、
- ー メモリ参照命令上に第2の状況情報を記憶するための複数のアドレス可能なデータ記憶要素を含む第2のデータ構造を規定する段階；
- ー 各々前記アドレス可能なデータ記憶要素の1つをポイントする、発行済みシリアル番号ポインタ (ISN)、完遂済みシリアル番号ポインタ (CSN)、最早期未解決予測制御転送命令シリアル番号ポインタ (PBSN)、非メモリ参照命令完遂済みシリアル番号ポインタ (NMCSN) を含む複数のポインタを規定する段階；
- ー 前記アドレス可能なデータ記憶要素のうちの予め定められたものを非固有的にポイントするよう前記複数のポインタの各々を初期

設定する段階；

- ー 前記命令発行ユニットから少なくとも1つの命令を発行する段階；
- ー 前記発行済み命令の各々がメモリを参照するか否かを決定する段階；

- ー 前記命令発行ユニットからの命令発行信号に応じて、メモリ参照命令及び非メモリ参照命令について活動中の命令を表示するように前記第1のデータ構造内の前記第1のインジケータをセットする段階、
- ー 非メモリ参照命令についてのみ活動中の命令を表示するよう前記第2のデータ構造内の前記第2のインジケータをセットし、前記命令発行ユニットからの命令発行信号に応じてメモリ参照命令についてこの命令が非活動中であることを表示するべく前記第2のインジケータをクリアする段階、
- ー (i) 命令が完了したこと及びこの完了にエラーが伴うか伴わないかを表示する前記実行ユニットからの命令完了信号、及び(ii) 予測された制御転送が適正に予測されたか否かを表わす予測評価状況ユニットからの制御転送命令のための予測評価状況信号、に応じて前記命令の現行状況を活動中又は非活動中として表示するように発行済みの各々の命令について前記第1及び第2のデータ構造内の前記第1及び第2のインジケータを各々変更することによって前記インジケータの状況を更新する段階；
- ー 最後の発行済み命令番号をポイントするべく、現行マシンサイクル中に発行された命令の数を表示する命令発行ユニットからの信号に応じて、前記SN ポインタを前進させる段階；
- ー 前記CSN ポインタから出発して増大するシリアル番号順で前記第1のインジケータを順序評価し、非活動中の命令インジケータを

もつ最後のシリアル番号を決定するべく前記ISN ポインタに向かって評価し、非活動中の命令インジケータをもつ前記最後のシリアル番号をポイントするべく前記CSN ポインタを前進させる段階；

- ー 予測評価状況ユニットからの予測評価状況信号に応じて最も早期の未解決分岐命令の場所をポイントするべく前記PBSNポインタを前進させる段階；
- ー 最後の完遂済み命令の場所をポイントするべく前記NMCSN ポインタを前進させ、さらに、前記第2のデータ構造内の前記第2のインジケータの評価に基づいて、前記PBSNポインタを上へあらゆる未完遂で活動中のメモリ命令を通過してCSNの前に前進させる段階；

— 予測制御転送命令の誤予測又は実行例外の発生の時点で精確な状態を回復することができるように、(i) 制御転送命令及び(ii) 投機的に修正できる制御レジスタ値を修正するという副作用をもつと考えられる命令である各々の命令を実行する直前にマシン状態情報を記憶する段階；及び

— 投機的に修正できない制御レジスタ値を修正するという副作用をもち得る命令を実行する直前に全ての保留命令を完遂し退去させるべくマシンを同期化する段階；

を含んで成り、かくして非メモリ参照命令に比べより多くのマシンサイクルを実行することを必要とするメモリ参照命令は、命令実行状況が許せば直ちに、ただし精確な状態を危険にさらすことなく、実行のためにスケジュールできるようになる方法。

49. 前記複数の規定されたポインタにはさらに、前記アドレス可能なデータ記憶要素のうちの1つをポイントする資源再生ポインタ(RRP)が含まれており、さらに

— 割振られた全てのマシン資源がマシンによって再生された、完遂されている最後の命令をポイントするべく、命令完遂及び再生ユ

ニット(ICRU)からの信号に応じて前記in-order RRPポインタを前進させる段階を含んで成る、請求項48に記載の方法。

50. 前記メモリ参照命令が、ロード命令、ストア命令及び原子命令を含むグループの中から選択される、請求項48に記載の方法。

51. 前記ISNポインタを前進させる前記段階が、以前のマシンサイクルの最後の発行済み命令シリアル番号のシリアル番号に基づいており、前記信号は、現マシンサイクル中に発行された命令の数を識別する請求項48に記載の方法。

52. 前記CSNポインタを前進させる前記段階は、各マシンサイクル上のシリアル番号の予め定められた数だけ、前記CSNポインタを前進させることに制限されている請求項48に記載の方法。

53. 前記制御転送命令には分岐命令が含まれている請求項48に記載の方法。

54. 内部データ記憶装置をもち外部メモリと連絡している投機的out-of-order

実行プロセッサの中で、精確な例外モデルを維持するロード及びストア命令を含めたメモリ参照命令をトラッキングし攻撃的にスケジュールするための方法において、

- ー 前記プロセッサ内での実行のため複数の命令を発行する段階；
 - ー 前記発行された命令のうちのどれに投機的実行が関与するかを識別し、各々の前記投機的命令に付随するインジケータを記憶する段階；
 - ー 前記発行済み命令のうちのどれが外部メモリを参照するかを識別し、各々の前記メモリ参照命令に付随するインジケータを記憶する段階；
 - ー 各命令について完了状況を決定するべくエラー無しの実行完了を監視することを含め、前記命令の発行後に前記複数の命令の実行状況を監視する段階；
 - ー 前記発行された命令の実行状況をトラッキングする段階；及び
 - ー 投機的命令であるものとしての識別を含むその他の命令の状況及び前記複数の命令の前記実行完了状況に基づいて、非メモリ参照命令に先立って実行するために前記発行済みのメモリ参照命令をスケジュールする段階、
- を含んで成る方法。

55. 内部のレジスタと外部のメモリを含むデータ記憶手段、命令発行ユニット及び命令実行ユニットを有する投機的out-of-order実行プロセッサの中で、このプロセッサ内の精確なアーキテクチャ状態を維持しながら、順序的にout-of-orderで低待ち時間命令に先立って、アーキテクチャ状態を修正できる長待ち時間命令を攻撃的にスケジュールするための装置において、

- ー エラー無く実行を完了し取消す必要が全く無い順序的に最後の連続的命令を識別するための手段；
- ー 実行の予測が誤った場合実行が取消されなくてはならない可能性のある順序的に最も早期の投機的発行済みで未解決の予測された制御転送命令を識別するための手段；
- ー エラー無しで実行を完了し取消す必要の全く無い前記識別された順序的に最後の連続的命令と、実行の予測を誤った場合取消さなければならない可能性のある前記識別された順序的に早期の投機的発行済みで未解決の予測された制御転送

命令の間に順序づけされたあらゆる長待ち時間命令を識別するための手段であって、この長待ち時間命令は、長待ち時間命令であるものとして予め指定された予め定められた命令セットと、発行済み命令を比較することによって識別される、識別用手段；

— 中間実行のための前記識別された長待ち時間命令をスケジュールするための手段；

を含んで成り、

かくして、誤予測をした可能性のある投機的に発行済みの介入する制御転送命令及びそうでなければ実行例外を生成した可能性のある命令とは無関係に、実行のためスケジュールされ得る長待ち時間命令のみを実行のためにスケジュールすることによって、長待ち時間命令が攻撃的に発行され精確な状態が維持されることになる、装置。

56. — n のアドレス可能場所 $0, 1, 2, \dots, n-1, n-1$ を有し、アドレス可能場所 $n-1$ がこのデータ記憶手段内で論理的にアドレス可能な場所 0 と隣接している、円形モジュロ n データ構造；

— 各命令が発行されるにつれて、各々の発行された命令に対して単調に増大する（モジュロ n ）数値的シリアル番号識別タグを割当てするための手段；

— 前記データ構造内の活動状況インジケータを前記タグに基づいて各々の発行済み命令と結びつけるための手段；

— その命令が発行された時点ですでに発行されていたことを表示するため、前記発行済み命令の各々全てについて前記データ構造内の前記 n 個の場所のうちの固有の 1 つの場所で前記活動状況インジケータの最初の要素をセットするための手段；

— 長待ち時間命令として識別された命令についてのみ、前記命令がその発行時点で長待ち時間命令であることを表示するべく、前記活動状況インジケータの第 2 の要素をセットするための手段；

— 予測された結果に基づいて投機的に発行された命令についてのみ、その命令がその発行時点で投機的に発行された命令であることを表示するべく、前記活動

状況インジケータの1要素をセットするための手段；

ー 前記命令がエラー無く完了した時点で、それがすでにエラー無く完了していたことを表示するべく、前記データ構造内の前記活動状況インジケータをクリアするための手段；

ー エラー無しで完了していた前記順序的に最後の連続的命令、前記順序的に最も早期の投機的未解決予測命令及び、前記識別された最後の連続的エラー無し完了済み命令と前記識別された早期未解決予測命令の間に順序づけされた前記長待ち時間命令を含む、発行済みの各々の命令についての実行状況を決定するべく、発行済みの複数の命令についての前記活動状況インジケータを評価することにより、発行済み命令の実行状況をトラッキングするための手段、
をさらに含んで成る請求項55に記載の装置。

57. エラー無しで完了した順序的に最後の連続的命令を識別するための前記段階には、前記データ構造内の1つのエントリをポイントし、最後の非活動化済み命令を表示する第2のポインタを設定するための手段が含まれており、前記最後の非活動化済み命令は、順序的に先行する全てのin-order命令もエラー無しで完了している、エラー無しで完了した順序的に最後のin-order命令であり、前記第2のポインタにより識別された命令よりも早い順序のin-order命令が、分岐誤予測及び例外とは無関係に実行可能であり；

ー その実行を取消さなければならない可能性のある順序的に最も早期の投機的で未解決予測制御転送命令を識別するための前記手段には、前記データ構造内の1つのエントリをポイントし、順序的に最も早期のin-order未解決分岐命令を表示する第1のポインタ（PBSN）を設定するための手段が含まれており、前記第1のポインタによって識別された命令より順序的に早期のin-order命令が分岐誤予測とは無関係に実行可能であり；

ー 前記識別された最後の連続的エラー無し完了命令と前記識別さ

れた最早期未解決予測制御転送命令の間に順序づけされたあらゆる長待ち時間命令を識別するための前記段階が、前記第2のポインタ（CSN）と前記第3のポイ

ンタ (NMCSN) の間のあらゆる長待ち時間命令を実行のためにスケジュールすべく利用可能なものとして識別するが、前記第2のポインタ (NMCSN) と前記第1のポインタ (PBSN) の間のあらゆる非長待ち時間命令を実行のためにスケジュールすべく利用可能なものとしては識別しないための手段を含んでいる、請求項56に記載の装置。

58. 前記識別用手段には、各マシンサイクル毎に前記状況インジケータ要素を論理的に比較するためのブール論理回路、及び前記ポインタの各々を、前記より高い数値的シリアル番号識別タグに対応する場所まで前進させるための手段が含まれている請求項57に記載の装置。

59. 前記長待ち時間命令には、前記外部メモリを参照する命令を含まれており、ロード命令及びストア命令を含むあらゆる長待ち時間命令を識別するための前記手段が、命令を復号し外部メモリ参照命令を識別するための命令デコーダを含んでいる請求項58に記載の装置。

60. 複数の内部プロセッサレジスタ及び外部メモリ、命令発行ユニット及び命令実行ユニットを有する投機的なout-of-order実行プロセッサの中で、このプロセッサ内の精確な状態を危険にさらすことなくレジスタのみを参照する命令に先立つアーキテクチャ状態を修正することのできるメモリ参照命令を攻撃的にスケジュールするための装置において；

ー メモリ参照命令及び非メモリ参照命令上に第1の状況情報を記憶するための複数のアドレス可能なデータ記憶要素を含む第1のデータ構造；

ー メモリ参照命令上に第2の状況情報を記憶するための複数のアドレス可能なデータ記憶要素を含む第2のデータ構造；

ー 各々前記アドレス可能なデータ記憶要素の1つをポイントする、発行済みシリアル番号ポインタ (ISN)、完遂済みシリアル番号ポインタ (CSN)、最早期未解決予測制御転送命令シリアル番号ポインタ (PBSN)、非メモリ参照命令完遂済みシリアル番号ポインタ (NMCSN) を含む前記内部プロセッサレジスタ内に構成され複数のポインタ値を記憶する複数のポインタ記憶レジスタ；

ー 前記アドレス可能なデータ記憶要素のうちの予め定められたものをポイント

するよう前記複数のポインタの各々を初期設定する手段；

- ー 前記命令発行ユニットから少なくとも1つの命令を発行する手段；
- ー 前記発行済み命令の各々が前記外部メモリを参照するか否かを決定する手段；
- ー 前記命令発行ユニットからの命令発行信号に応じて、メモリ参照命令及び非メモリ参照命令について活動中の命令を表示するように前記第1のデータ構造内の前記第1のインジケータをセットする手段；
- ー 非メモリ参照命令についてのみ活動中の命令を表示するよう前記第2のデータ構造内の前記第2のインジケータをセットし、前記命令発行ユニットからの命令発行信号に応じてメモリ参照命令についてこの命令が非活動中であることを表示するべく前記第2のインジケータをクリアする手段；
- ー (i) 命令が完了したこと及びこの完了にエラーが伴うか伴わないかを表示する前記実行ユニットからの命令完了信号、及び(ii) 予測された制御転送が適正に予測されたか否かを表わす予測評価

状況ユニットからの制御転送命令のための予測評価状況信号、に応じて前記命令の現行状況を活動中又は非活動中として表示するように発行済みの各々の命令について前記第1及び第2のデータ構造内の前記第1及び第2のインジケータを各々変更することによって前記インジケータの状況を更新する手段；

- ー 最後の発行済み命令番号をポイントするべく、現行マシンサイクル中に発行された命令の数を表示する命令発行ユニットからの信号に応じて、前記ISN ポインタを前進させる手段；
- ー 前記CSN ポインタから出発して増大するシリアル番号順で前記第1のインジケータを順序評価し、非活動中の命令インジケータをもつ最後のシリアル番号を決定するべく前記ISN ポインタに向かって評価し、非活動中の命令インジケータをもつ前記最後のシリアル番号をポイントするべく前記CSN ポインタを前進させる手段
- ー 予測評価状況ユニットからの予測評価状況信号に応じて最も早期の未解決分岐命令の場所をポイントするべく前記PBSNポインタを前進させる手段；

ー 最後の完遂済み命令の場所をポイントするべく前記MCSN ポインタを前進させ、さらに、前記第2のデータ構造内の前記第2のインジケータの評価に基づいて、前記PBSNポインタを上へあらゆる未完遂で活動中のメモリ命令を通過してCSNの前に前進させる手段；

ー 予測制御転送命令の誤予測又は実行例外の発生の時点で精確な状態を回復することができるように、(i) 制御転送命令及び(ii) 投機的に修正できる制御レジスタ値を修正するという副作用をもつと考えられる命令である各々の命令を実行する直前にマシン状態情報を記憶する手段；及び

ー 投機的に修正できない制御レジスタ値を修正するという副作用をもち得る命令を実行する直前に全ての保留命令を完遂し退去させ

るべく前記プロセッサを同期化する手段；

を含んで成り、かくして非メモリ参照命令に比べより多くのマシンサイクルを実行することを必要とするメモリ参照命令は、命令実行状況が許せば直ちに、ただし精確な状態を危険にさらすことなく、実行のためにスケジュールできるようになる装置。

61. 前記複数の規定されたポイントにはさらに、前記アドレス可能なデータ記憶要素のうちの1つをポイントする資源再生ポイント(RRP)が含まれており、さらに

ー 割振られた全てのマシン資源がマシンによって再生された完遂されている最後の命令をポイントするべく、命令完遂及び再生ユニット(ICRU)からの信号に応じて前記in-order RRPポインタを前進させる手段を含んで成る、請求項60に記載の方法。

62. 中央処理ユニット(CPU)についての精確なアーキテクチャ状態を維持しながらチェックポイント実行済み状態を低減させるべくこのCPU内の命令をチェックポイント実行するための方法において、

ー 前記CPU内で実行されたときにアーキテクチャ状態を修正する可能性のある命令を、その発行及び実行前に予め識別する段階；

ー 予め定められた選択基準に基づいて実行する前にアーキテクチャ状態をチェ

ックポイント実行せずに特別な実行モードにて実行するための、前記識別された命令のうちの特定の命令を予め選択する段階；

- ー 実行に先立ち前記予め選択された特定の命令以外の前記識別された命令についてアーキテクチャ状態をチェックポイント実行する段階；及び
- ー 前記特殊モードでの前記命令のうちの特定の命令の実行を含め、前記識別された命令を実行する段階、

を含んで成る方法。

63. 前記予め定められた基準には、修正可能な状態のタイプ及び実行中に命令によって修正され得る修正可能状態の量、が含まれる、請求項62に記載の方法。

64. 前記予め定められた基準には、予め定められた名目命令ストリーム内で命令が発生する頻度がさらに含まれている、請求項63に記載の方法。

65. 前記予め定められた基準には、さらに、前記CPU 内の特定のタイプの処理タスクについて前記選択基準を最適化できるように前記予め定められた名目命令ストリームの組成が含まれている請求項64に記載の方法。

66. 前記予め定められた基準には、

- ー 前記予め識別された命令のうちの各々の特定の命令によって修正可能な状態；
 - ー 前記予め識別された命令のうちの各々の特定の命令の修正可能な状態についての記憶必要条件；及び、
 - ー いくつかの予め定められた名目処理タスクについて前記予め識別された命令のうちの各々の特定の命令が発生する推定上の統計的頻度；
- が含まれている請求項62に記載の方法。

67. 前記予備選択段階には、

- ー 前記特別な実行モードでの実行のために、予め定められた低い推定上の統計的頻度で発行され比較的多いチェックポイント記憶量を必要とするような予め識別された命令を選ぶ段階；及び
- ー 正規の処理モードで、予め定められた高い頻度で発行され比較的小さいチェックポイント記憶量を必要とするような予め識別された命令を選ぶ段階；

が含まれている、請求項66に記載の方法。

68. 前記特別な実行モードには、前記選択された命令の実行を開始する前に前記CPUを同期化する段階が含まれている請求項67に記載の方法。

69. 前記CPU同期化段階には、

- ー 実行前に命令ストリーム内でマシンの同期化を必要とする同期化命令として、1つの命令を識別する段階；
- ー 実行がエラー無く完了し実行結果が状態ライトバックされてしまうように全ての保留中の発行済み命令が完遂され退去されるまで、前記同期化命令の実行を遅延させる段階；
- ー マシンの同期化を必要とする命令を逐次的かつin-orderで実行する段階；
- ー 状態ライトバックが行なわれる前に、前記同期化命令の各々の実行から生じる例外条件を識別する段階；
- ー 前記同期化命令の実行中に生じ得る前記例外条件のいずれかをとり扱う段階；及び
- ー 実行結果をマシン状態へライトバックする段階；

が含まれており、かくして前記同期化命令をチェックポイント実行することなく精確な状態が維持される、請求項68に記載の方法。

70. 1つの命令を同期化命令として識別する前記段階には、その命令を発行前に同期化命令として識別することが含まれており、前記同期化命令の実行を遅延させる前記段階にはさらに、全ての保留中の発行済み命令が完遂され退去されてしまうまで前記同期化命令の発行を遅延させることが含まれている請求項68に記載の方法。

71. 前記識別された命令には、予測されたプログラム制御転送命令、プログラムフローを修正できる命令、マシン状態を修正できる命令及び、制御レジスタ値を修正する副作用をもち得る命令、が含

まれる請求項68に記載の方法。

72. 前記プログラムフローを修正しうる命令にはBPr, FBcc, FBPcc, Bcc, BPcc, CALL, JMPL, TCC, RETURN, DONE, RETRY、及びそれらの組合せから成るグル

ープの中から選択されたSPARC-V9命令セット内で実施された命令が含まれている請求項71に記載の方法。

73. 前記マシン状態を修正できる命令には、制御レジスタ内で制御レジスタに書込む命令、及びレジスタウインドウ制御命令が含まれている請求項71に記載の方法。

74. 前記マシン状態を修正できる命令には、制御レジスタ内で制御レジスタを読取る命令が含まれている請求項71に記載の方法。

75. 前記識別された命令にはさらに、往々にして発行トラップを結果としてもたらず命令が含まれている請求項71に記載の方法。

76. 前記識別された命令にはさらに、トラップが投機的にとられた場合に復帰を行なうことができるように、トラップハンドルーチン内の第1命令が含まれている請求項71に記載の方法。

77. 前記CPU についての精確なアーキテクチャ状態を維持しながらチェックポイント実行済み状態を低減させるべく命令をチェックポイント実行するための方法において、

- ー 前記CPU 内で実行されたときにアーキテクチャ状態を修正する可能性のある命令を、その発行及び実行前に予め識別する段階；

- ー 予め定められた選択基準に基づいて実行する前にアーキテクチャ状態をチェックポイント実行せずにマシン同期化モードにて実行するための、前記識別された命令のうちの特定の命令を予め選択する段階；

を含み、

前記予め定められた基準には、

- ー 前記予め識別された命令のうちの各々の特定の命令によって修

正可能である状態；

- ー 前記予め識別された命令のうちの各々の特定の命令の修正可能な状態についての記憶必要条件；

- ー いくつかの予め識別された名目処理タスクについて前記予め識別された命令のうちの各々の特定の命令が発生する推定上の統計的頻度、

が含まれており、

前記予備選択段階には、

ー 前記特別な実行モードでの実行のために、予め定められた低い推定上の統計的頻度で発行され比較的多いチェックポイント記憶量を必要とするような予め識別された命令を選ぶ段階；及び、

ー 正規の処理モードで、予め定められた高い頻度で発行され比較的少ないチェックポイント記憶量を必要とするような予め識別された命令を選ぶ段階；

が含まれており、

さらには、

ー 実行に先立ち前記予め定め選択された特定の命令以外の前記識別された命令についてアーキテクチャ状態をチェックポイント実行する段階；及び

ー 前記マシン同期化モードでの前記命令のうちの特定の命令の実行を含め、前記識別された命令を、それらが前記命令ストリーム内に発生するにつれて実行する段階；

を含んで成り、

前記マシン同期化モードにはさらに、

ー 発行前に命令ストリーム内でマシンの同期化を必要とする同期化命令として、1 の命令を識別する段階；

ー 前記同期化命令以外の命令を発行し、保留中の発行済み命令が

エラー無しで実行を完了しこの実行からの結果が状態ライトバックされてしまうまで前記同期化命令の実行を遅延させる段階；

ー マシン同期化を必要とする各々の命令を逐次的かつin-orderで発行する段階；

ー 状態ライトバックが行なわれる前に、前記同期化命令の各々の実行から生じる例外条件を識別する段階；

ー 前記同期化命令の実行中に生じ得る前記例外条件のいずれかをとり扱う段階；及び

ー 実行結果を状態ライトバックする段階；

が含まれており、かくして前記同期化命令をチェックポイント実行することなく
精確な状態が維持される、方法。

78. 命令状況をトラッキングし、この命令状況に基づいてチェックポイントを
割振り及び割振り解除するための方法において、

- ー 前記CPU のメモリ内のデータ記憶領域内に1つのデータ構造を設定する段階
；

- ー 前記データ構造内で、複数のチェックポイント状況フィールドを有し単一の
チェックポイントに各々対応している複数のチェックポイント割振りレジスタを
規定する段階；

- ー 命令シリアル番号SNをもつ命令のために次の逐次的に利用可能なチェックポ
イント割振りレジスタnを割振り、前記複数の状況フィールド内で前記命令のた
めのチェックポイント状況情報を記憶する段階、

- ー 完遂済みの命令シリアル番号ポインタ (CSN) が、チェックポイントn及び
次の記憶されたチェックポイントn+1に付随する命令シリアル番号SNを通過し
たか否かを決定するため、予め定められた時間的間隔で、完遂済みシリアル番号
ポインタ (CSN) を監視する段階；

- ー CPU のバックアップ信号に応答して、前記CPU に対して、前記チェックポ
イント割振りレジスタ内に記憶された情報を伝達する段階；及び

- ー 完遂済み命令シリアル番号ポインタ (CSN) がチェックポイントn及び次の
チェックポイントn+1に付随する命令シリアル番号を通過したという決定に応
答して、以前に割振られたチェックポイントを割振り解除する段階、
を含んで成る方法。

79. CPU が、チェックポイントが形成されたチェックポイントシリアル番号を
通過してバックアップしたことを表わすCPU バックアップ信号の受理に応答して
、以前に割振られたチェックポイントを割振り解除する段階をさらに含んで成る
請求項78に記載の方法。

80. 前記データ構造内で、複数のチェックポイント状況フィールドをもち単一
のチェックポイントに各々対応する複数のチェックポイント割振りレジスタを構

成する前記段階には、

ー 各々の前記チェックポイントレジスタで、そのチェックポイントが割振りされたか及びその特定のチェックポイント内に記憶された情報が有効なチェックポイント情報かを指示する第1の有効性状況フィールドを規定する段階；

ー 各々の前記チェックポイントレジスタで、完遂済みシリアル番号命令ポイント（CSN）がレジスタ内に記憶された特定のチェックポイントの命令シリアル番号（SN）を通過したか否かを表示することによってチェックポイントが通過されたか否かを指示する第2のチェックポイント通過済み状況フィールドを規定する段階；

ー 各々の前記チェックポイントレジスタで、完遂済み命令シリアル番号ポイント（CSN）に付随する条件を指示する第3の状況フィールドを規定する段階；

ー 各々の前記チェックポイントレジスタで、完遂済み命令シリアル番号ポイント（CSN）がチェックポイントのシリアル番号よりも大きいか否かを指示し、完遂済みシリアル番号がこのチェックポイントを通過した時点を見極めるために用いられる、第4の比較的大きい状況フィールドを規定する段階；

ー 各々の前記チェックポイントレジスタで、チェックポイント実行されたシリアル番号の後の次のシリアル番号（NSN）を指示する第5の次シリアル番号状況フィールドを規定する段階、

を含んで成る請求項78に記載の方法。

81. 前記第1の状況フィールドが1ビットのフィールドであり、前記第2の状況フィールドが1ビットのフィールドであり、前記第3の状況フィールドが3ビットのフィールドであり、前記第4の状況フィールドが1ビットのフィールドであり、前記第5の状況フィールドが6ビットのフィールドである、請求項80に記載の方法。

82. 命令シリアル番号SNをもつ命令についてチェックポイント*i*を割振る前記段階には、

ー 次の利用可能なチェックポイント番号*i*を識別し、この番号*i*に基づいて前記チェックポイント割振りレジスタ*i*の中へ指標づけする段階；

- ー 前記チェックポイント i が有効であり退去されていないことを表示するべく前記レジスタの前記第 1 のフィールドに「1」を書込む段階；
- ー 前記チェックポイント i を通過していないことを表示するため前記第 2 のフィールド内に「0」を書込む段階；
- ー 円形の活動中の命令状況リングのまわりからの前記完遂済み命令シリアル番号ポインタ (CSN) の遷移を検出することができるように、前記チェックポイント i が割振られた時点で前記完遂済み命令

シリアル番号ポインタ (CSN) の場所を表示するべく前記第 3 のフィールド内に複数の状態のうちの 1 つを書込む段階；

- ー チェックポイントのシリアル番号と完遂済みシリアル番号ポインタ (CSN) を比較し、完遂済みシリアル番号の方が大きい場合、完遂済みシリアル番号がこのチェックポイントを通過したことを表示するため、前記第 4 の状況フィールド内にインジケータを書込む段階；及び
- ー 第 5 の状況フィールド内に、そのチェックポイントが形成された命令シリアル番号に関する情報を書込む段階、
が含まれている請求項 80 に記載の方法。

83. 前記命令シリアル番号に関する情報は、そのチェックポイントが形成されたシリアル番号である、請求項 80 に記載の方法。

84. 前記命令シリアル番号に関する情報は、そのチェックポイントが形成されたシリアル番号に 1 シリアル番号を増分した番号である請求項 8 に記載の方法。

85. 完遂済み命令シリアル番号ポインタ (CSN) がチェックポイント n 及びチェックポイント n + 1 に付随する命令シリアル番号を通過したという決定にตอบสนองして以前に割振られたチェックポイントを割振り解除する前記段階にはさらに、

- ー チェックポイントレジスタ内に記録されたシリアル番号と完遂済みシリアル番号ポインタの間の差を計算することによって完遂済みシリアル番号通過済み信号を生成する段階；及び

- ー 前記生成された完遂済みシリアル番号通過済み信号にตอบสนองして前記第 1 の状

況フィールドをクリアする段階、
が含まれている請求項82に記載の方法。

86. — CPU バックアップ表示信号；

— CPU が、チェックポイントが形成された前記バックアップ表示

信号の中で識別されたチェックポイントシリアル番号を通過してバックアップしたか否かを決定する段階；及び

— 前記決定が、バックアップがそのチェックポイントを通過して行なわれたことを識別した場合、このチェックポイントを割振り解除する段階；

をさらに含んで成る請求項78に記載の方法。

87. — バックアップチェックポイントでの命令のシリアル番号を識別するCPU バックアップ表示信号を検出する段階；

— チェックポイントが形成された前記バックアップ表示信号内で識別されたチェックポイントシリアル番号を通過してCPU がバックアップしたか否かを決定する段階；及び

— 前記生成された完遂済みシリアル番号通過済み信号に応じて、前記第1の状況フィールドをクリアすることによってバックアップがチェックポイントを通過してであることを前記決定が識別した場合に、前記チェックポイントを割振り解除する段階、

をさらに含んで成る請求項85に記載の方法。

88. — 最後に発行された命令シリアル番号を表示する発行済みシリアル番号 (ISN) 信号を受理する段階；

— 最後に完遂された命令シリアル番号を表示する完遂済み命令シリアル番号 (CSN) 信号を受理する段階；

— 完遂済みシリアル番号ポインタ (CSN) とバックアップチェックポイントの間のシリアル番号を有する全ての発行済み命令をキルする段階；

— バックアップチェックポイントに付随する次のシリアル番号を記憶する前記第5の状況フィールドに基づいて前記バックアップ中に、発行済み命令シリアル番号ポインタ (ISN) を更新する段階；

- ー 未終了の浮動小数点例外又はデータ区切り点例外が検出された

場合にバックアップチェックポイントのためにチェックポイントレジスタ内に記憶された次のシリアル番号値に基づいて発行済みシリアル番号ポインタ (ISN) を更新する段階；

をさらに含んで成る請求項85に記載の方法。

89. チェックポイントのシリアル番号と完遂済みシリアル番号ポインタ (CSN) を比較する前記段階が、複数の前記チェックポイントレジスタについて同時に行なわれる請求項82に記載の方法。

90. 前記チェックポイントは、最後のチェックポイントが形成されてから予め定められた数の事象が発生したこと及び命令タイプ属性とは無関係に付加的なチェックポイントが形成されるべきであることを表示するタイムアウトチェックポイント割振り信号に応答して前記チェックポイントが割振られる請求項78に記載の方法。

91. 前記命令の実行に先立って前記CPU を同期化することによってCPU 内に正確な状態を維持するための方法において、

- ー 実行前に命令ストリーム内でマシンの同期化を必要とする同期化命令として、実行中にマシン状態を修正し得る1つの命令を識別する段階；
- ー 前記保留中の命令の実行がエラー無く完了し実行結果がCPU 状態ライトバックされてしまうように全ての保留中の発行済み命令が完遂され退去されるまで、前記同期化命令の実行を遅延させる段階；
- ー マシンの同期化を必要とする命令を逐次的かつin-orderで実行する段階；
- ー 状態ライトバックが行なわれる前に、前記同期化命令の各々の実行から生じる例外条件を識別する段階；
- ー 前記同期化命令の実行中に生じ得る前記例外条件のいずれかをとり扱う段階；及び

- ー 実行結果をマシン状態へライトバックする段階；

を含んで成り、かくして前記同期化命令によって修正可能な状態を予め記憶する

ことなく前記同期化命令について精確な状態が維持されることになる、方法。

92. 1つの命令を同期化命令として識別する前記段階には、その命令を発行前に同期化命令として識別することが含まれており、前記同期化命令の実行を遅延させる前記段階にはさらに、全ての保留中の発行済み命令が完遂され退去されてしまうまで前記同期化命令の発行を遅延させることが含まれている請求項91に記載の方法。

93. CPU のための最適なチェックポイントサイズを設計する方法において、

- ー 前記CPU により支持されている命令セット内の各命令により修正可能なマシン状態を決定する段階；
- ー いくつかの代表的な目処処理タスクについて各々の特定の命令が発生する統計的頻度を決定する段階；
- ー 各命令について状態記憶必要条件を決定する段階；
- ー チェックポイント情報のための最大記憶量を選ぶ段階；
- ー 稀にしか発行されず、より頻繁に発生する命令に比べ比較的大きい量のチェックポイント記憶を必要とするような命令を選択するため最適化を行なう段階；
- ー 前記割り振りされた最大チェックポイントスペース内でチェックポイント実行できるような命令のみをチェックポイント実行する段階；

を含んで成る方法。

94. 中央処理ユニット (CPU) 内の精確な状態を維持しながらチェックポイント実行されたデータの量を減少させるべく 1つの命令をチェックポイント実行するための方法において、

- ー 前記中央処理ユニット内のデータ記憶装置内に複数のマップレジスタ要素をもつレジスタリネームマップデータ構造を規定する段階；
- ー 1つの命令の実行に先立ちその命令が参照した各々の論理ソース及び／又は論理宛先レジスタのために前記CPU 内の利用可能な物理レジスタを割り振る段階；
- ー 固有の論理レジスタタグにより、各々の前記論理ソースレジスタを識別する段階；
- ー 物理レジスタタグにより各々の割り振り可能な物理レジスタを識別する段階；

- ー 前記固有の論理レジスタタグのうちの 1 つと前記複数のマップレジスタ要素のうちの各々 1 つを結びつける段階；及び
 - ー 各々のマップレジスタ要素が現在その論理レジスタタグに対してマッピングされている物理レジスタタグを記憶するような形で、この物理レジスタが割振られた固有の論理レジスタタグに付随するマップレジスタ要素の中に各々の前記割振られた物理レジスタで識別された物理レジスタタグを記憶する段階；
- を含んで成り、

かくして、前記命令の発行の結果として前記レジスタのために変化した全ての状態は記憶され、CPU バックアップが必要とされる場合に回復のため利用可能な状態になる、方法。

95. ー 前記 CPU 内の利用可能な未割振り物理レジスタについて 1 つずつの物理レジスタを各々記憶している複数のフリーリスト要素をもつフリー物理レジスタフリーリストデータ構造を規定する段階；

- ー 論理レジスタを参照する命令がディスパッチされそうであるという命令発行ユニットからの表示に応じて、ソース又は宛先レジス

タとして論理レジスタを置換するのに使用するため命令発行ユニットに対しフリー物理レジスタタグを伝達する段階；及び

- ー 前記命令発行ユニットに対する前記フリー物理レジスタタグの伝達時点で、前記フリー物理レジスタフリーリストデータ構造から前記物理レジスタタグを除去する段階

をさらに含んで成る、請求項 94 に記載の方法。

96. ー データを記憶するための論理レジスタタグフィールド、旧物理レジスタタグフィールド及び再生要素妥当性フィールドを各々有する複数の再生要素をもつ資源再生ユニットを提供する段階；

- ー 前記再生要素を固有の発行済み命令シリアル番号と結びつける段階；及び
- ー 特定の論理レジスタを参照する命令シリアル番号 SN の割当てられた命令の復号に応じて：

- ・ 前記複数のリネームマップレジスタ要素のうちのどの特定の 1 つの要素が、

該当する場所、以前に復号され発行された命令について前記復号された命令の中で参照された前記特定の論理レジスタと結びつけられたかを決定する段階、

- ・前記特定の論理レジスタに付随する特定の物理レジスタタグを決定するための前記特定のリネームマップレジスタ要素を読取る段階、

- ・前記古い物理レジスタタグフィールドの中に前記特定の物理レジスタタグを記憶し、前記命令シリアル番号SNに付随する再生要素の前記論理レジスタタグフィールドの中に前記特定の論理レジスタタグを記録して、旧物理レジスタ番号が以前にマッピングした論理レジスタタグと共に記憶されるようにする段階；

- ・特定の再生要素が有効なデータセットを含有することを表示するため、前記命令シリアル番号SNに付随する再生要素妥当性フィー

ルド内に1つのインジケータを書込む段階；

- ・命令シリアル番号SNをもつ前記命令を実行する上で使用するため前記フリーリストから新しいフリー物理レジスタを識別し、前記フリー物理レジスタに対応する新しいフリー物理レジスタタグを中に書込む段階；

- ・前記命令シリアル番号SNをもつ命令により参照された前記特定の論理レジスタについて新しい物理レジスタを割振る段階；及び

- ・各々のリネームマップレジスタ要素が、現在特定の論理レジスタにマッピングされている物理レジスタタグを記憶するような形で前記論理レジスタを参照する命令シリアル番号SNに応じて前記新しい物理レジスタが割振られた論理レジスタタグに付随するリネームマップレジスタ要素内に前記新たに割振られた物理レジスタに対応する前記新しい物理レジスタタグを記憶する段階；

をさらに含んで成り、

かくして、前記資源再生ユニットは、必要とあらばより早期の論理－物理マッピングを再構築する元となる逐次履歴レジャーとして役立つことができる、請求項95に記載の方法。

97. 前記レジスタリネームマップデータ構造を規定する段階には、前記CPU内の複数のレジスタタイプの各々について別々のレジスタリネームマップデータ構造を規定する段階が含まれており；前記フリー物理レジスタを規定する段階には

、前記CPU 内の複数のレジスタタイプの各々について別々のフリーリストデータ構造を規定する段階が含まれており；前記資源再生ユニットを提供する段階には、前記複数の別々のレジスタリネームデータ構造及び別々のフリーリストデータ構造のうちの任意のものによりマッピングされた論理及び物理レジスタを記憶する単一の資源再生ユニットを提供する段階が含まれている請求項96に記載の方法。

98. — バックステップすべき命令の数の表示を含むCPU バックステップ開始信号を受理する段階；

— 最後の発行済み命令シリアル番号を表示する、発行済み命令シリアル番号（ISN）を受理する段階；

— 前記受理された発行済み命令シリアル番号及び各再生要素と固有の発行済み命令シリアル番号の前記結びつきに基づいて特定の再生要素を識別する段階；

— 各々のリネームマップレジスタ要素が前記命令ISN の発行に先立ち特定の論理レジスタにマッピングされた物理レジスタタグを記憶するように、前記発行済み命令シリアル番号（ISN）に付随する前記資源再生要素の中に記憶されたデータに基づいて命令シリアル番号ISN のための前記識別された特定のレジスタリネームマップ要素の中に記憶された論理レジスタから物理レジスタへのマッピングを回復する段階、及び

— 発行済み命令シリアル番号を減分し、前記特定の再生要素の識別段階及び、バックステップされるべき各命令のための識別された要素内に記憶された論理レジスタタグ—物理レジスタマッピングの前記回復段階をくり返す段階、をさらに含んで成る、請求項96に記載の方法。

99. 前記各々の命令シリアル番号において、論理レジスタから論理レジスタへのマッピングを回復する段階には、

— 命令シリアル番号により前記資源再生データ構造へと逆方向に指標付けし、前記命令シリアル番号に付随する前記旧物理レジスタタグフィールド内に記憶された前記物理レジスタタグ及び前記論理レジスタタグフィールド内に記憶された論理レジスタタグを読取る段階、

ー 前記論理レジスタタグフィールドから読みとられた前記論理レ

ジスタタグに付随するレジスタリネームマップ要素を識別する段階；

ー 中に記憶された物理レジスタタグを決定するべく前記論理レジスタタグに対応するリネームマップ要素を読取る段階；

ー 前記物理レジスタフリーリスト内へ前記リネームマップ要素内に記憶された前記決定された物理レジスタタグを書込む段階；

ー 前記資源再生ユニットの前記論理レジスタタグフィールドの中に記憶された前記論理レジスタタグに対応するレジスタリネームマップ要素の中へ前記旧物理レジスタタグフィールド内に記憶された前記物理レジスタタグを書込む段階、及び

ー 再生要素が有効なデータセットを含まないことを表示するべく前記命令シリアル番号のための前記再生要素妥当性フィールド内へインジケータを書込む段階；

が含まれており、

かくして前記物理レジスタフリーリストユニット要素、前記レジスタリネームマップ要素及び前記レジスタ再生ユニット要素が、前記特定の命令の発行の直前に有していた状態まで回復されることになる請求項98に記載の方法。

100. ー 資源退去ポインタ（RRP）として記憶された命令シリアル番号を決定する段階；

ー 前記資源退去ポインタ（RRP）が特定の記憶された再生要素に付随する命令シリアル番号SN以上となるよう前進させられた時点で命令シリアル番号SNのための特別な論理レジスタに以前に割振られた物理レジスタの割振りを解放する段階、

をさらに含んで成る請求項99に記載の方法。

101. 前記物理レジスタの割振りを解放する段階には、

ー 前記物理レジスタフリーリストの中へ、前記命令シリアル番号

に対応する前記特別なりネームマップ要素の中に記憶された前記物理レジスタ

グを書込む段階；及び

ー 再生要素が有効なデータセットを含まないことを表示するべく前記命令シリアル番号のための前記再生要素妥当性フィールド内へ1つのインジケータを書込む段階、

が含まれている、請求項100に記載の方法。

102. 前記命令の実行に先立ち物理レジスタを割振る前記段階には、実行ユニットに前記命令発行する前に前記物理レジスタを割振る段階が含まれている請求項94に記載の方法。

103. 前記複数のレジスタタイプが、整数レジスタタイプ、浮動小数点レジスタタイプ、条件コードレジスタタイプ、Y-レジスタタイプ、トラップスタックレジスタタイプから成るグループの中から選択される請求項97に記載の方法。

104. 物理レジスタの数が論理レジスタの数よりも多く、各々の前記論理レジスタを複数の物理レジスタ上へとマッピングすることができる請求項94に記載の方法。

105. 命令により参照される各々の論理レジスタは、任意の論理宛先レジスタが物理レジスタへとマッピングされる前に物理レジスタマッピングされる、請求項94に記載の方法。

106. 構築されたレジスタ場所を物理レジスタ場所へマッピングすることにより命令が発行された時点で使用されたレジスタについてのレジスタリネームを実施するCPUにおいて、精確な状態を維持しながらチェックポイント実行されたデータの量を減少させるための方法において、

- ー 前記発行済み命令により使用される論理ソースレジスタを識別する段階；
- ー 利用可能な物理レジスタ上へと各々の識別された論理ソースレ

ジスタをマッピングし、資源リネームデータ構造内に前記論理ー物理マッピングを記憶する段階；

- ー 前記発行済み命令によって使用される論理宛先レジスタを識別する段階；
- ー 利用可能な物理レジスタ上に各々の識別された論理宛先レジスタをマッピングし、資源リネームデータ構造内に前記論理ー物理マッピングを記憶する段階；

- ー フリー物理レジスタのリストを記憶するために前記CPU 内に第1のデータ記憶装置を提供する段階；
 - ー 命令シリアル番号SN、命令シリアル番号SNをもつ前記命令により参照された論理レジスタのための論理レジスタタグ及び前記論理レジスタにマッピングされた物理レジスタのための物理レジスタタグを記憶するため、前記CPU 内に第2のデータ記憶装置を提供する段階；
 - ー 命令実行に先立つ命令発行の間、命令実行の前に発行済み命令の影響を受ける前記リネームソース及び／又は宛先レジスタの状態に関するマシン状態データを記憶することによって前記リネームされた論理及び物理レジスタをチェックポイント実行する段階；
- を含んで成る方法。

107. ー CPU バックアップ開始信号条件の受理に応答して、前記リネームマップを、バックアップを結果としてもたらした命令の実行の直前にそれが有していた状態まで回復させることにより、より早期のマシン状態を回復する段階をさらに含んで成る請求項106 に記載の方法。

108. 命令を発行するための命令発行ユニット、及び命令を実行するための実行ユニットを有する、in-orderで発行された命令のout-of-order投機的実行を実施する中央処理ユニットの中で、

- ー 前記CPU 内で実行された時点でアーキテクチャ状態を修正する可能性があり、実行に先立ちプロセッサの状態をセーブするべくチェックポイント実行されるべき第1の命令タイプセットのアイデンティティを予備記憶するための記憶手段；
- ー 前記CPU 内で実行された時点でアーキテクチャ状態を修正する可能性があり、実行に先立つアーキテクチャ状態をチェックポイント実行することなく特別な実行モードで実行されるべき第2の命令タイプセットのアイデンティティを予備記憶するための記憶手段；
- ー 前記第1及び第2の記憶された命令タイプと発行された命令タイプを比較し、発行された命令が、正規実行モードでは実行前にチェックポイント実行される

べきであり、特殊なプロセッサモードではチェックポイント実行無しに実行されるべきであることを表示する出力信号を生成するための比較器；

ー 実行に先立ち前記第1のタイプの命令についてアーキテクチャ状態をチェックポイント実行するべくチェックポイント形成を開始するため、前記比較器出力信号に応答するチェックポイント手段；及び

ー 前記第2のタイプの命令を前記特殊な実行モードで実行することを含めた、前記命令を実行するための命令実行手段、

を含む装置において、前記特殊な実行モードには、前記第2のタイプの命令を実行する前に前記プロセッサを同期化するための手段が含まれている装置。

109. 命令を発行するための命令発行ユニット、データを記憶するためのメモリ及び命令を実行するための実行ユニットを有する、in-orderで発行された命令のout-of-order投機的実行を実施する中央処理ユニットの中で、

ー 前記CPUのメモリ内のデータ記憶領域内のデータ構造；

ー 前記データ構造内で各々対応する単一のチェックポイントに割振られ、複数のチェックポイント状況フィールドを有する複数のチェックポイント割振りレジスタ；

ー 命令シリアル番号SNをもつ命令のために次の逐次的に利用可能なチェックポイント割振りレジスタnを割振るための手段；

ー 前記複数の状況フィールド内で前記命令のためのチェックポイント状況情報を記憶するための手段；

ー 完遂済み命令シリアル番号ポインタ(CSN)がチェックポイントn及び次の記憶されたチェックポイントn+1に付随する命令シリアル番号SNを通過したか否かを決定するため予め定められた時間的間隔で完遂済みシリアル番号ポインタ(CSN)を監視するための手段；

ー CPUバックアップ信号に応答して前記チェックポイント割振りレジスタ内に記憶された情報を前記CPUに伝達するための手段；及び

ー 完遂済み命令シリアル番号ポインタ(CSN)がチェックポイントn及び次のチェックポイントn+1に付随する命令シリアル番号を通過したという決定に応

答して、以前に割振られたチェックポイントを割振り解除するための手段；
を含んで成り、

かくして命令状況がトラッキングされ、命令状況に基づいてチェックポイントが割振りされ割振り解除されることになる、装置。

110. 命令を発行するための命令発行ユニット、データを記憶するためのメモリ及び命令を実行するための実行ユニットを有する、in-orderで発行された命令のout-of-order投機的実行を実施する中央処理ユニットの中で、選択された命令の実行に先立ち前記CPUを同期化するための装置において、

- ー 実行前に命令ストリーム内でマシンの同期化を必要とする同期化命令として、実行中にマシン状態を修正する可能性のある1つの命令を識別するための手段；
 - ー 前記保留中の命令の実行がエラー無く完了し実行結果がCPU状態にライトバックされてしまうように全ての保留中の発行済み命令が完遂され退去されるまで、前記同期化命令の実行を遅延させるための手段；
 - ー マシンの同期化を必要とする命令を逐次的かつin-orderで実行するための、手段；
 - ー 状態ライトバックが行なわれる前に、前記同期化命令の各々の実行から生じる例外条件を識別するための手段；
 - ー 前記同期化命令の実行中に生じ得る前記例外条件のいずれかをとり扱うための例外ハンドラー；及び
 - ー 実行結果をマシン状態へライトバックするための手段、
- を含んで成り、

かくして、前記同期化命令により修正可能な状態を予備記憶することなく、前記同期化命令のために精確な状態が維持されることになる、装置。

111. 命令を発行するための命令発行ユニット、データを記憶するためのメモリ及び命令を実行するための実行ユニットを有する、in-orderで発行された命令のout-of-order投機的実行を実施する中央処理ユニットの中で、このユニットの中の精確な状態を維持しながらチェックポイント実行されたデータの量を減少さ

せるべく1つの命令をチェックポイント実行するための装置において、

- ー 前記中央処理ユニット内のデータ記憶装置内に複数のマップレジスタ要素をもつレジスタリネームマップデータ構造；
- ー 1つの命令の実行に先立ちその命令が参照した各々の論理ソー

ス及び／又は論理宛先レジスタのために前記CPU内の利用可能な物理レジスタを割振るための手段

- ー 固有の論理レジスタタグにより、各々の前記論理ソースレジスタを識別するための手段；
- ー 物理レジスタタグにより各々の割振り可能な物理レジスタを識別するための手段；
- ー 前記固有の論理レジスタタグのうちの1つと前記複数のマップレジスタ要素のうちの各々1つを結びつけるための手段；及び
- ー 各々のマップレジスタ要素が現在その論理レジスタタグに対しマッピングされている物理レジスタタグを記憶するような形で、この物理レジスタが割振られた固有の論理レジスタタグに付随するマップレジスタ要素の中に各々の前記割振られた物理レジスタで識別された物理レジスタタグを記憶するための手段を含んで成り、

かくして、前記命令の発行の結果として前記レジスタのために変化した全ての状態は記憶され、CPUバックアップが必要とされる場合に回復のため利用可能な状態になる、装置。

112. 命令発行手段、命令実行手段及びデータを記憶するためのデータ記憶装置を有する中央処理ユニット（CPU）の中で、前記実行手段の中で発生する例外のために例外取扱いを束縛するための方法において、

- ー タイムアウト条件を規定する特定された事象の発生回数について予め定められた閾値を設定し、この閾値を前記CPU内の第1のデータ記憶装置の中に記憶する段階、
- ー 前記CPU内のカウンタ内で前記特定された事象の発生を計数し、前記発生回数を前記CPU内の第2のデータ記憶装置内に1つの計数として記憶する段階；

- 前記計数を前記閾値と比較する段階；
 - 前記計数が前記タイムアウト条件以上である場合にタイムアウトチェックポイントを形成する段階、
- を含んで成る方法。

113. — 第1の命令の実行に先立って前記計数を初期値に初期設定する段階；及び
- 前記タイムアウトチェックポイントが形成された後を含め、チェックポイントの形成後に前記計数を前記初期状態にリセットする段階、
- をさらに含んで成る、請求項112に記載の方法。

114. 前記特定された事象は、前記中央処理ユニットによるあらゆる命令の発行であり、かくして前記タイムアウトチェックポイントの形成は、連続的に形成されたチェックポイント間で発行される命令の数を前記予め定められた命令数に制限するようになっている、請求項113に記載の方法。

115. 前記命令が任意の命令属性をもつ、請求項114に記載の方法。

116. 前記特定された事象が、予め定められた数のマシクロックサイクルの実行である、請求項113に記載の方法。

117. 前記特定された事象が、チェックポイントを形成することなく通過した命令発行サイクルの数である、請求項114に記載の方法。

118. 前記タイムアウトチェックポイントを形成する前記段階は、前記CPU内の全てのレジスタに関する状態情報を記憶することを含め、前記CPUの完全な状態を特徴づける状態情報を記憶することを含む、請求項114に記載の方法。

119. 前記タイムアウトチェックポイントを形成する前記段階に

は、最後のタイムアウトチェックポイントを含む最後のチェックポイントが前記CPU内に形成されてから変化したデータの徴候のみを記憶するレジスタリネームマップの状態を特徴づける状態情報を記憶することが含まれている、請求項114に記載の方法。

120. 前記徴候には、物理レジスタに対する論理レジスタのマッピングが含まれている、請求項119に記載の方法。

121. ー 障害発生命令よりも順序的に前のチェックポイント実行された命令のために形成されたチェックポイントから、より早期のプロセッサ状態を回復する段階；

ー 前記障害発生命令から前記より早期のチェックポイント実行済み命令へと前記CPU をバックアップし、前記チェックポイント実行済み命令から順方向に命令を再度実行する段階、

を含んで成る、障害発生命令によって引き起こされた実行エラーからの回復段階をさらに含んで成る請求項114 に記載の方法。

122. ー 前記障害発生命令より順序的に後のチェックポイント実行済み命令のために形成されたチェックポイントから、より早期のプロセッサ状態を回復する段階；

ー 前記障害発生命令から前記順序的に後のチェックポイント実行済み命令へと前記CPU をバックアップする段階；

ー 逆方向の段階的やり方で、順序的に前記より後のチェックポイント実行済み命令と前記障害発生命令の間に存在する各々の命令のためにプロセッサ状態を回復することを含め、前記障害発生命令へと前記CPU をバックステップさせる段階；及び

ー 前記障害発生命令から順方向に命令を再度実行する段階；

を含む、障害発生命令によって引き起こされた実行エラーからの回復段階をさらに含んで成る請求項114 に記載の方法。

123. ー タイムアウトチェックポイントが満了した時点で全て

のチェックポイントが割振られた場合、命令復号タイプのチェックポイントのための精確な状態を維持するのに必要となり得るようにマシンをマシン停止させるのではなく予め定められた規則に従ってタイムアウトチェックポイントの形成を遅延させる段階、

をさらに含んで成る請求項114 に記載の方法。

124. 予め定められた規則には、予め定められたサイクル数についてのタイムアウトチェックポイントの形成を遅延させることが含まれている、請求項23 に

記載の方法。

125. 命令を発行するための命令発行ユニットをもつ中央処理ユニット (CPU) の中で例外取扱いを束縛するための方法において、

- ー 前記CPU 内のデータ記憶装置内で予め定められたタイムアウト閾値計数を設定する段階；
- ー 前記CPU 内のデータ記憶装置内に初期カウント数を記憶する段階；
- ー 1つの命令が発行されたCPU クロックサイクル中に、少なくとも1つの命令が発行されたことを表示するべく、このクロックサイクル中に前記命令発行ユニットから命令発行済み信号を送る段階；
- ー 前記命令発行済み信号に応答して、受理した命令発行済み信号の数を計数するためのタイムアウトカウンタを提供する段階；
- ー 前記命令発行済み信号を受理する段階；
- ー 各CPU クロックサイクル中に受理された命令発行済み信号の数を計数し、前記CPU 内のデータ記憶装置内に記憶された累積数にその数を加算する段階；
- ー タイムアウトチェックポイント作成信号に応答して形成されたチェックポイント及び復号済み命令属性に応答して形成されたチェックポイントを含め、チェックポイントが形成された時点でつねに前記カウント数を初期値にリセットする段階；
- ー 各々のCPU クロックサイクルの間に、前記累積カウント数が閾値計数以上であるか否かを決定するため前記タイムアウト閾値計数と前記累積カウント数を比較する段階；
- ー 前記タイムアウトカウンタ内で、前記累積カウント数が前記閾値計数以上である場合に、タイムアウトチェックポイント作成信号を生成する段階；
- ー 前記タイムアウトチェックポイント信号を前記命令発行ユニットに伝送し、チェックポイントタイムアウト条件が発生しチェックポイントを1つ形成しなければならないということを前記発行ユニットに知らせる段階；
- ー 前記タイムアウトチェックポイント作成信号に応じてチェックポイントを形成し、前記初期値に前記カウント数を取りセットする段階、

を含んで成り、

かくして前記タイムアウトチェックポイントの形成が、連続的に形成されたチェックポイントの間で発行された命令の数を予め定められた命令数に制限しかくして、例外条件が発生した時点で取消しされ再実行されなくてはならない可能性のある発行済み命令の数を束縛することになる、方法。

126. 前記累積カウント数を前記タイムアウト閾値計数と比較する前記段階の後でかつ前記生成段階の前に；さらに

ー 同じCPU クロックサイクル内で発行されなくてはならない命令発行ウィンドウ内の命令がチェックポイントを必要としているか否かを、命令属性に基づいて決定する段階；

ー 前記タイムアウトチェックポイント作成信号の生成を遅延させる段階、

ー 前記復号済み命令に応答してチェックポイントを形成する段階

；及び

ー 命令属性に基づくチェックポイント形成のため、チェックポイント形成に回答して前記初期値チェックポイントに前記累積カウント数をリセットする段階；
を含んで成る請求項125 に記載の方法。

127. 命令の発行時点で命令発行済み信号を生成する命令発行ユニット、前記命令の実行に先立ちプロセッサ状態を記憶する、チェックポイントの形成時点でチェックポイント形成済み信号を生成するチェックポイント形成ユニットをもつプロセッサの中で、

ー 前記命令発行済み信号を受理し、各々の受理した命令発行済み信号に応じて計数を増分するための有効化入力ポート、前記チェックポイント形成済み信号及び累積計数出力信号を受理するためのリセットポートを有するカウンタ回路；

ー 予め定められたタイムアウト閾値計数を記憶するためのデータ記憶装置；及び

ー 前記記憶されたタイムアウト閾値計数を前記累積計数出力信号と比較するための比較器、

を含んで成り、

前記カウンタ回路は、命令発行済み信号の予め定められた数を計数した後タイムアウトチェックポイント要求信号を出力する、
タイムアウトチェックポイントカウンタ装置。

128. 命令を発行するための発行ユニット、命令を実行するための実行ユニット及び中央処理ユニット内のその他のユニットに対し前記実行ユニットからの実行結果を伝達するデータ順方向分配バスを有する中央処理ユニットの中で、複数の投機的に発行された予測された命令の実行結果を同時に監視するための方法において、

ー プロセッサ内の前記データ順方向分配全体にわたり前記実行ユ

ニットからの実行結果信号を受理するように結合されたウォッチポイントデータを記憶するための複数のウォッチポイントレジスタをもつウォッチポイントユニットを提供する段階；

ー 前記投機的に発行された予測された命令の各々について、制御フロー転送方向を左右し投機的発行済み予測命令を投機的に発行する基礎となる1つの条件の予測値を識別する予測条件データ結果を含むウォッチポイントデータを記憶するために1つのウォッチポイントレジスタを割振る段階；

ー 前記データ順方向送りバス上で伝送される前記投機的発行済みの予測命令についての実行結果信号を監視する段階；

ー 前記記憶されたウォッチポイントデータ及び実際の既知の条件データ結果信号及び予め定められた規則を含む前記記憶されたウォッチポイントデータ及び実行結果信号に基づき、予め定められた事象の発生を検出する段階であって、ここで前記実際の既知の条件データ結果信号が、前記投機的発行済みの予測命令の制御フロー転送方向を決定する上で基礎となるべき条件の実際の値を識別しているような、段階；

ー 信号が一致しているかしていないかを決定するべく前記投機的に発行された命令に関して前記データ順方向送りバス上で到着した前記結果信号と前記投機的に発行された予測命令のうちの1つについて前記ウォッチポイントレジスタ内に記憶された前記ウォッチポイントデータを比較する段階であって、一致は前記投

機能的発行済み予測命令が正しく予測されていたことを表わし、不一致は、前記投機的発行済み予測命令が該予測されていたことを表わすような、段階；及び

－ 比較の結果、前記予測が誤予測であったことが表示された場合には、前記誤予測に基づいて実行された命令が取消されるように前

記中央処理ユニットをより早期の中央処理ユニット状態まで回復させる段階、を含んで成る方法。

129. － 前記データ順方向送りバス上に現われる実行結果信号がもはや前記予め定められた事象の発生を検知する結果をもたらさないように前記中央処理ユニットを回復させる段階の後に前記ウォッチポイント要素を割振り解除する段階；

をさらに含んで成る、請求項128 に記載の方法。

130. 前記中央処理ユニットを回復させる段階には、

－ 前記実際の既知の条件データ結果信号がフェッチされ誤予測を訂正するべく発行され得るように、適正な命令ストリームに対応する前記発行ユニットに対する新しい命令アドレスを供給する段階；

－ 前記の直前の状態まで中央処理ユニットのマシン状態を回復し、前記誤予測に基づいて実行された命令が取消されるように命令の実行を再開する段階；

－ 前記フェッチされた命令に基づいて前記適正な命令ストリーム内で前記新しい命令のための実行を開始する段階、

が含まれている、請求項128 に記載の方法。

131. 前記投機的発行済み予測命令の実行に先立って、チェックポイントデータ記憶装置内に中央処理ユニットマシン状態を記憶する段階をさらに含んで成り、

前記中央処理ユニット状態の回復段階には、前記新しい命令のための前記実行開始段階に先立って前記チェックポイントデータ記憶装置から中央処理ユニット状態を回復することが含まれている請求項130 に記載の方法。

132. 前記中央処理ユニットをより早期の中央処理ユニット状態まで回復させる前記段階にはさらに、前記より早期の状態まで回復

させるべく前記中央処理ユニットをバックアップすることが含まれている請求項131に記載の方法。

133. 前記中央処理ユニットをより早期の中央処理ユニット状態まで回復させる前記段階にはさらに、前記より早期の状態まで回復させるべく前記中央処理ユニットをバックアップしかつバックステップすることが含まれている請求項131に記載の方法。

134. 前記投機的発行済み予測命令のうちの1つのための前記ウォッチポイントの中に記憶された前記ウォッチポイントデータがWP_COND 信号を含み；前記データ順方向送りバス上で到着した前記結果信号には、XCC_DATA_C, ICC_DATA_C又はFCC_DATA_C, FXU_XCC_DATA, FXU_ICC_DATA, FXAGU_XCC_DATA, FXU_ICC_DATA、及びFPU_FCCDATAから成るグループの中から選択された実行ユニット条件コードデータが含まれている、請求項128に記載の方法。

135. 前記命令には、予測された分岐命令が含まれている請求項128に記載の方法。

136. 前記命令にはジャンプ&リンク命令が含まれている請求項128に記載の方法。

137. 前記命令には、予測された分岐及びジャンプ&リンク命令が含まれている請求項128に記載の方法。

138. 前記新しいアドレスからウォッチポイントユニット内のデータ記憶装置内に記憶されている請求項128に記載の方法。

139. 命令を発行するための命令発行ユニット、発行された命令を実行するための実行ユニット及び前記実行ユニットからCPUのその他のコンポーネントまで実行結果を伝達するための複数のデータ順方向送りバスを有する中央処理ユニット（CPU）の中で、前記実行結果生成ユニットから予め定められた命令が待っている実行結果を同時に捕捉するための方法において、

ー 前記結果データを捕捉することのできる各々の実行結果ソースから、有効化制御信号を受理する段階；

ー 各々の前記実行結果ソースユニットから結果データタイプ信号を受理する段

階；

- － 前記実行結果ソースユニットのうちのどれから前記結果データが受理されることになるかを表示するセレクトソース信号及び前記結果データが予測されている現行のクロックサイクルとの関係における時刻の表示を受理する段階；
 - － 選択された実行結果ソースユニットを規定するべく予め定められた規則に従って前記制御有効化信号、前記結果データタイプ信号及び前記セレクトソース信号の受理に応答してデータ捕捉有効化信号を生成する段階；
 - － 前記実行結果ソースユニットから実行結果データ信号を監視する段階；
 - － 前記到達した実行結果データ及び前記データ捕捉有効化信号に応答して前記選択された実行結果ソースユニットから実行結果データを捕捉する段階；及び
 - － 前記中央処理ユニット内のデータ記憶装置内での評価のため前記捕捉された実行結果データを記憶する段階、
- を含んで成る方法。

140. 少なくとも1つのデータ捕捉有効化信号の生成に応答して前記捕捉された実行結果データの評価を開始する段階、

をさらに含んで成る請求項139に記載の方法。

141. ー 抑止制御信号を受理する段階；及び
- － 前記抑止選択信号が受理された場合に前記データ捕捉有効化信号を生成する前記段階を抑止する段階；
- をさらに含んで成る請求項139に記載の方法。

142. 前記抑止制御信号には、現行のマシンサイクルにて発行された命令によって条件コードが修正されたこと、又現行サイクル中に前記データ順方向送りバス上に現われうる条件コードが現行サイクル中有効でない可能性があるということを表わす前記条件コードリネームユニットからの条件コードリネーム済み信号が含まれている請求項141に記載の方法。

143. 前記実行結果データには条件コードデータが含まれている請求項139に記載の方法。

144. 前記実行結果データには、リネームされた条件コードデータが含まれて

いる請求項139 に記載の方法。

145. 前記実行結果データには、浮動小数点実行ユニット、整数実行ユニット及びアドレス生成ユニットから受理された条件コードデータ及び条件コードリネームユニットから受理されたリネームされた条件コードデータが含まれており、これらの条件コードデータ及び前記リネームされた条件コードデータが同じマシンサイクル中に受理されている請求項139 に記載の方法。

146. 前記実行結果データには条件コードデータが含まれ、前記予め定められた命令には、予測された分岐命令を含む予測された制御フロー転送命令が含まれている請求項139 に記載の方法。

147. 前記実行結果データには、計算されたジャンプ&リンクアドレスが含まれており、前記予め定められた命令には、ジャンプ&リンク命令が含まれている請求項139 に記載の方法。

148. 前記実行結果データには条件コードデータ及び計算上のジャンプ&リンクアドレスが含まれており、前記予め定められた命令には予測された分岐命令及びジャンプ&リンク命令が含まれている、請求項139 に記載の方法。

149. 前記実行結果ソースユニットが命令実行ユニットを含んで

いる請求項139 に記載の方法。

150. 前記実行結果ソースユニットにはさらにレジスタリネームユニットが含まれる請求項139 に記載の方法。

151. 前記レジスタリネームユニットには条件コードレジスタリネームユニットが含まれ、前記実行計算データにはリネーム条件コードデータが含まれている、請求項150 に記載の方法。

152. 前記実行結果ソースユニットには整数実行ユニット、浮動小数点実行ユニット、アドレス生成ユニット及び条件コードレジスタリネームユニットが含まれている請求項139 に記載の方法。

153. 前記結果データが条件コードデータ及びアドレスデータを含んでいる請求項139 に記載の方法。

154. 前記条件コードタイプの信号には、その命令が発行された時点で生成さ

れた前記条件コードリネームユニットからの条件コードタイプの信号、及び命令発行中により早期に生成されその後ウォッチポイント記憶要素の中に記憶された条件コードタイプの信号が含まれている請求項139 に記載の方法。

155. BR_XCC信号、BR_ICC信号、BR_FCC信号及びその組合せから成るグループの中から前記条件コードタイプ信号が選択される請求項154 に記載の方法。

156. 前記条件コードタイプ信号がWP_XCC信号、WP_ICC信号、WP_FCC信号及びその組合せから成るグループの中から選択されている請求項154 に記載の方法。

157. — 前記有効化制御信号がDO_PREDICT_VEC又はWP_ACTIVEから成るグループの中から選択され

— 前記データタイプ信号が、BR_XCC信号、BR_ICC信号、BR_FCC信号、WP_XCC信号、WP_ICC信号及びWP_FCC信号から成るグループの中から選択され；

— 前記レクトソース信号は、CC_DV_C, FXU_CC_CURR_MATCH, FXUCC_ARRAY_MATCH, FXAGU_CC_CURR_MATCH, FXAGU_CC_ARRAY_MATCH, FPU_CC_CURR_MATCH、及びFPU_CC_ARRAY_MATCHから成るグループの中から選択され；

— 前記生成されたソース選択信号は、XCC_DATA_C, FXU_DATA_C or ICC_DATA_C, FXU_XCC_DATA_F, FXAGU_CC_DATA_F, FXAGU_CC_DATA_F、及びFPU_CC_DATA_F信号、から成るグループの中から選択されたソース信号を選択するべくSEL_BR_XCC, SEL_BR_ICC, SEL_FXU_XCC, SEL_FXU_ICC, SEL_FXAGU_XCC, SEL_FXAGU_ICC, SEL_FPU_FCCから成るグループの中から選択され；

— 前記捕捉された実行結果データは、XCC_DATA_C, FXU_DATA_C or ICC_DATA_C, FXU_XCC_DATA_F, FXAGU_CC_DATA_F, FXAGU_CC_DATA_F, FPU_CC_DATA_Fデータ信号から成るグループの中から選択され、

— 評価のための前記捕捉された実行結果データがEVAL_CC信号を含んでいる、請求項139 に記載の方法。

158. 前記データ記憶ユニットがラッチである請求項139 に記載の方法。

159. 前記複数の実行結果ソースユニットには、整数実行ユニット、アドレス生成ユニット、浮動小数点実行ユニット及び条件コードリネームユニットが含まれている請求項139 に記載の方法。

160. 前記有効化制御信号にはDO_PREDICT_VEC信号が含まれ；前記有効化制御信号にはWP_ACTIVE信号が含まれ；前記実行結果データタイプ信号には条件コードタイプ信号が含まれている請求項139 に記載の方法。

161. 命令を発行するための命令発行ユニット、命令を実行するための実行ユニット及び前記中央処理ユニットのその他のコンポー

ネントに対して実行結果を伝達するための複数のデータ順方向送りバスを有する中央処理ユニットの中で、現行のCPU クロックサイクル内でリネームすることにより修正されなかった条件コードタグについて前記データ順方向送りバスから条件コードタグを捕捉するための方法において、

- ー 現行マシンサイクル内で発生する条件コードタグの修正を除外するものの、リネームされた条件コード物理レジスタタグを識別するリネームされた条件コード物理レジスタタグ信号を受理する段階；

- ー 前記データ順方向送りバスから直接複数の条件コードタグ信号を受理する段階；

- ー 前記受理したリネーム済み条件コードタグ信号を前記実行ユニットから到着した前記複数の条件コードタグ信号のうちの全てのものと同時に比較し、前記到着したデータ順方向送りバス条件コードタグ信号の各々に対応する比較信号を生成する段階；

- ー 現行CPU サイクル中に前記データ順方向送りバス上で有効なデータを送る各々の実行ユニットからデータ有効信号を受理する段階；

- ー 各々の前記比較信号を、現行CPU サイクル中に前記データ順方向送りバスの上で有効データを送る各々の実行ユニットから到着した異なるデータ有効制御信号の論理積を同時にとる段階；

- ー 各々の比較信号及び各々の前記データ有効信号が表明された時点で条件コード現行一致信号を生成する段階；

を含んで成り、

- ー 前記条件コード現行一致信号は、前記ウォッチポイント番号のいずれであるかを識別しかくして現行のCPU サイクル内で前記データ順方向送りバスから捕捉

され得る付随する命令番号及び条件コードタグを識別し、

－ かくして、前記現行一致論理は、未解決の投機的に発行された複数の命令を同時に監視しかつ、捕捉して前記投機的発行済み命令の発行の基礎となった予測の評価のために利用できるようにすることのできる条件コードを識別することが可能となる、方法。

162. 命令を発行するための命令発行ユニット、命令を実行するための実行ユニット論理及び物理トリネームするためのレジスタリネームユニット及び前記中央処理ユニットのその他のコンポーネントに対して前記実行ユニットからの実行結果を伝達するための複数のデータ順方向送りバスを有する中央処理ユニットの中で、現行のCPU クロックサイクル内でリネームすることにより修正されなかった条件コードタグについて前記データ順送りバスから条件コードタグを捕捉するための方法において

－ 前記CPU 内にデータを記憶するため複数の条件コードタグレジスタをもつ条件コードタグアレイデータ記憶装置を提供する段階；

－ 現行マシンサイクル内で発生する条件コードの修正を含めてリネームされた条件コード物理レジスタタグを識別するリネームされた条件コード物理レジスタタグ信号を受理する段階；

－ ウォッチポイント番号及びそのCPU サイクル中に発行された各々の投機的発行済み命令に割振られた対応する条件コードタグレジスタ要素を識別する信号ウォッチポイント番号信号成分を含む予測実施信号を受理する段階；

－ そのとき現行であるCPU サイクル内で発行された命令によって1つの条件コードが修正された時点で条件コードリネーム済み信号を受理し、この条件コードリネーム済み信号を、前記予測実施信号によって識別された条件コードタグレジスタ要素内への前記リネーム済み条件コード物理レジスタタグ信号の書込みのための書込み有効化信号として用いる段階；

－ 論理積をとることによって、前記条件コードリネーム済み信号と前記予測実施信号を組合せ、前記条件コードリネーム済み信号と前記予測実施（信号）の両

方が表明された時点でのみ前記タグアレイに対する前記受理されたりネーム済み条件コード物理レジスタタグの書込み及びこの書込みのこのアレイを伴う場所を制御するべく書込み制御有効化信号を生成する段階；

ー アレイ中の特定の要素が前記予測実施信号の一成分として含まれているウォッチポイント番号により識別され指標付けされている、前記書込み有効化信号の制御下で前記タグアレイ内の前記複数の条件コードタグレジスタ要素の1つの中に前記投機的発行済み命令に付随する前記受理されたりネーム済み条件コード物理レジスタタグ信号を記憶する段階；

ー 前記リネーム済み条件コード物理レジスタタグが前記データ順方向送りバス上で到着する条件コードタグ信号と一致するの必要がなくなってしまうまで、前記条件コードタグレジスタ内に前記記憶された各々のリネーム済み条件コード物理レジスタタグ信号を保持する段階；

ー 同じCPU サイクル中に前記データ順方向送りバスからの複数の条件コードタグ信号を同時に受理する段階；

ー 複数のアレイ比較信号を生成するべく前記記憶されたりネーム済み条件コード物理レジスタタグ信号の全てのものと前記受理されたデータ順方向送りバス条件コード信号の各々を同時に比較する段階；及び

ー 前記データ有効信号と前記アレイ比較信号の論理積をとり、前記比較信号と前記データ有効信号が両方共表明された時点で条件コードアレイ一致信号を生成する段階；

を含んで成り、

ー 前記条件コードアレイ一致信号は、前記ウォッチポイント番号のいずれであるかを識別しかくして現行のサイクル内で前記データ順方向送りバスから捕捉され得る付随する命令番号及び条件コードタグを識別し、

ー かくして、前記アレイ晚期一致論理は、未解決の投機的に発行された複数の命令を同時に監視しかつ、捕捉して前記投機的発行済み命令の発行の基礎となった予測の評価のために利用できるようにすることのできる条件コードを識別することが可能となる、方法。

163. — 現行マシンサイクル内で発生する条件コードタグの修正を除外するものの、リネームされた条件コード物理レジスタタグを識別するリネームされた条件コード物理レジスタタグ信号を受理する段階；

— 前記データ順方向送りバスから直接複数の条件コードタグ信号を受理する段階；

— 前記受理したリネーム済み条件コードタグ信号を前記実行ユニットから到着した前記複数の条件コードタグ信号のうちの全てのものと同時に比較し、前記到着したデータ順方向送りバス条件コードタグ信号の各々に対応する比較信号を生成する段階；

— 現行CPU サイクル中に前記データ順方向送りバス上で有効なデータを送る各々の実行ユニットからデータ有効信号を受理する段階；

— 各々の前記比較信号を、現行CPU サイクル中に前記データ順方向送りバスの上で有効データを送る各々の実行ユニットから到着した異なるデータ有効制御信号の論理積を同時にとる段階；

— 各々の比較信号及び各々の前記データ有効信号が表明された時点で条件コード現行一致信号を生成する段階；

をさらに含んで成り、

— 前記条件コード現行一致信号は、前記ウォッチポイント番号のいずれであるかを識別しかくして現行のCPU サイクル内で前記データ順方向送りバスから捕捉され得る付随する命令番号及び条件コードタグを識別し、

— かくして、前記現行一致論理は、未解決の投機的に発行された複数の命令を同時に監視しかつ、捕捉して前記投機的発行済み命令の発行の基礎となった予測の評価のために利用できるようにすることのできる条件コードを識別することが可能となる、請求項139 に記載の方法。

164. — 前記CPU 内にデータを記憶するため複数の条件コードタグレジスタをもつ条件コードタグアレイデータ記憶装置を提供する段階；

— 現行マシンサイクル内で発生する条件コードの修正を含めてリネームされた条件コード物理レジスタタグを識別するリネームされた条件コード物理レジスタ

タグ信号を受理する段階；

ー ウォッチポイント番号及びそのCPU サイクル中に発行された各々の投機的発行済み命令に割振られた対応する条件コードタグレジスタ要素を識別する信号ウォッチポイント番号信号成分を含む予測実施信号を受理する段階；

ー そのとき現行であるCPU サイクル内で発行された命令によって1つの条件コードが修正された時点で条件コードリネーム済み信号を受理し、この条件コードリネーム済み信号を、前記予測実施信号によって識別された条件コードタグレジスタ要素内への前記リネーム済み条件コード物理レジスタタグ信号の書込みのための書込み有効化信号として用いる段階；

ー 論理積をとることによって、前記条件コードリネーム済み信号と前記予測実施信号を組合せ、前記条件コードリネーム済み信号と

前記予測実施（信号）の両方が表明された時点でのみ前記タグアレイに対する前記受理されたりネーム済み条件コード物理レジスタタグの書込み及びこの書込みのこのアレイを伴う場所を制御するべく書込み制御有効化信号を生成する段階；

ー アレイ中の特定の要素が前記予測実施信号の一成分として含まれているウォッチポイント番号により識別され指標付けされている、前記書込み有効化信号の制御下で前記タグアレイ内の前記複数の条件コードタグレジスタ要素の1つの中に前記投機的発行済み命令に付随する前記受理されたりネーム済み条件コード物理レジスタタグ信号を記憶する段階；

ー 前記リネーム済み条件コード物理レジスタタグが前記データ順方向送りバス上で到着する条件コードタグ信号と一致する必要がなくなってしまうまで、前記条件コードタグレジスタ内に前記記憶された各々のリネーム済み条件コード物理レジスタタグ信号を保持する段階；

ー 同じCPU サイクル中に前記データ順方向送りバスからの複数の条件コードタグ信号を同時に受理する段階；

ー 複数のアレイ比較信号を生成するべく前記記憶されたりネーム済み条件コード物理レジスタタグ信号の全てのものと前記受理されたデータ順方向送りバス条件コード信号の各々を同時に比較する段階；及び

ー 前記データ有効信号と前記アレイ比較信号の論理積をとり、前記比較信号と前記データ有効信号が両方共表明された時点で条件コードアレイ一致信号を生成する段階；

をさらに含んで成り、

ー 前記条件コードアレイ一致信号は、前記ウォッチポイント番号のいずれであるかを識別しかくして現行のサイクル内で前記データ

順方向送りバスから捕捉され得る付随する命令番号及び条件コードタグを識別し、

ー かくして、前記アレイ晚期一致論理は、未解決の投機的に発行された複数の命令を同時に監視しかつ、捕捉して前記投機的発行済み命令の発行の基礎となった予測の評価のために利用できるようにすることのできる条件コードを識別することが可能となる、請求項163 に記載の方法。

165. ー 現行の発行ウインドウ信号内の現在リネームされた条件コードタグを各々のタグ順方向送りバスタグに比較し、先行するサイクルの条件コードタグを各々のデータ順方向送りバスタグと比較することにより、現行の発行ウインドウサイクル内の現在リネームされている条件コードと先行サイクル中の条件コードタグの間でデータ順方向送りバスタグ比較を分離する段階

を含む、比較時間を低減するべく前記条件コードデータと前記データ順方向送りバスタグを比較することをさらに含んで成り、

ー かくして先行するサイクル内の前記条件コードタグ及び現在リネームされている条件コードタグとの前記データ順方向送りバスタグの前記比較は、それが単一の非分割作業にて行なわれた場合に比べさらに迅速に行なわれることになる、請求項164 に記載の方法。

166. 命令を発行するための発行ユニット、命令を実行するための実行ユニット及び中央処理ユニット内で前記実行ユニットからその他のユニットまで実行結果を伝達するための手段、を有する中央処理ユニット (CPU) の中で、複数の投機的発行済み命令の実行結果を同時に監視するための方法において、

ー 命令識別タグで各々の命令を識別する段階；

ー 各々の投機的発行済み命令について命令発行中に前記命令識別タグに付随する指標付けされたデータ記憶装置の中に前記CPU 内の

予測データを記憶し、この予測データを前記投機的発行済み命令と結びつける段階であって、前記予測データがその命令が投機的に発行された条件の予測値を識別している、段階；

ー 前記実行結果伝達手段上で伝送された全ての発行済み命令について、実行結果信号を同時に監視する段階；

ー 前記実行結果データが前記伝達用手段上に現われた時点で前記予測データに対応するものの必ずしも等しいものではない既知のデータを含む前記実行結果データのうちの選択されたデータを同時に捕捉する段階であって、前記既知のデータが、前記命令を投機的に発行する目的で仮定された条件の実際値を識別している、段階；

ー 前記予め定められたデータを既知のデータと比較し、この予め定められたデータが前記既知のデータと一致しない場合に誤予測信号を生成する段階；及び

ー 前記誤予測に基づいて実行された命令が取消されるように、誤予測信号の受理に応じてより早期の中央処理ユニット状態まで中央処理ユニットを回復させる段階、

を含んで成る方法。

167. 中央処理ユニットの中で投機的に発行された予測された分岐命令を含む予め定められた投機的発行済み命令のための複数の誤予測信号を同時に生成するための方法において、

ー 前記投機的発行済み命令を監視するためにウォッチポイントが形成されつつあることを表示するウォッチポイント活動中信号を受理する段階；

ー 前記投機的発行済み命令の各々について前記CPU 内に規定されたデータ記憶構造の中でウォッチポイント番号及び付随するウォッチポイントレジスタを割振りする段階；

ー 前記1つの投機的発行済み命令に関係するウォッチポイントデ

ータを受理し、前記割振られたウォッチポイントレジスタ内に前記ウォッチポイントデータを記憶する段階；

ー 各々の発行済み分岐命令について、分岐命令のための複数の誤予測信号を同時に生成する段階であって、

- ・条件コード選択論理回路から評価条件コード信号を受理する段階；
- ・ウォッチポイント記憶要素からウォッチポイント条件信号を受理する段階；
- ・ウォッチポイント記憶要素から条件コードタイプ制御信号を受理する段階；
- ・各実行結果ソースユニットから有効化制御信号を受理する段階；
- ・分岐のための条件が正しく予測されているか否かを評価するため各々の前記実行結果ソースユニットのための前記有効化制御信号の表明された状態を含め予め定められた状態コード評価規則に従って前記ウォッチポイント条件信号と前記評価条件コード信号を比較する段階；

・前記分岐命令が正しく予測された場合には、分岐真信号を表明する段階；
を含む段階；及び

ー 前記分岐真信号が表明されたという決定に応答して評価真信号を生成する段階；

ー 投機的発行済み分岐命令について前記結果データを捕捉した結果として少なくとも1つのデータ捕捉有効化信号の受理に応じて評価準備完了信号を生成する段階；

ー 前記評価準備完了信号と前記評価真信号の両方の受理に応じて命令誤予測信号を生成する段階、

を含んで成る方法。

168. 中央処理ユニットの中で、投機的に発行されたジャンプ&リンク命令を含む予め定められた投機的発行済み命令のための複数の誤予測信号を当時に生成するための方法において、

ー 前記投機的発行済み命令を監視するためにウォッチポイントが形成されつつあることを表示するウォッチポイント活動中信号を受理する段階；

ー 前記投機的発行済み命令の各々について前記CPU内に規定されたデータ記憶

構造の中でウォッチポイント番号及び付随するウォッチポイントレジスタを割振りする段階；

ー ウォッチポイントがジャンプ&リンク命令のために形成されつつあることを表示するウォッチポイントジャンプ&リンク有効化信号を受理する段階；

ー 前記1つの投機的発行済み命令に関係するウォッチポイントデータを受理し、前記割振られたウォッチポイントレジスタ内に前記ウォッチポイントデータを記憶する段階；

ー 各々の前記発行済みジャンプ&リンク命令について、複数のジャンプ&リンク誤予測信号を同時に生成する段階であって、

・プログラムカウンタデータを記憶するための複数の記憶場所をもつ標的アドレスデータ構造を規定する段階；

・前記ジャンプ&リンク命令が発行された時点で前記標的アドレスデータ構造記憶場所のうちの1つに代替的な次のプログラムカウンタを書込み、この記憶された代替的な次のプログラムカウンタを前記命令と結びつける段階；

・実行結果ソースユニット内での前記ジャンプ&リンクの実行中に計算上の次のプログラムカウンタ値を計算する段階；

・前記ジャンプ&リンク命令の実行の完了時点でジャンプ&リン

ク命令実行完了信号を生成する段階；

・前記実行結果ソースユニットから前記計算上の次のプログラムカウンタを受理する段階；

・前記記憶場所から前記ジャンプ&リンク命令に付随する前記記憶された代替的な次のカウンタを読取る段階；

・前記代替的な次のプログラムカウンタ値を前記計算上の次のプログラムカウンタ値と比較する段階；

・前記代替的な次のプログラムカウンタ値が前記計算上の次のプログラムカウンタ値と一致する場合、ジャンプ&リンク一致信号を生成する段階；及び

・前記ジャンプ&リンク一致信号と前記ウォッチポイントジャンプ&リンク有効化信号の両方の表明に応じてジャンプ&リンク真信号を生成する段階、

を含む段階；

ー 前記ジャンプ&リンク一致信号が表明されたという決定に応答して評価真信号を生成する段階；

ー 前記ウォッチポイント活動中信号、前記命令ウォッチポイント番号信号、前記ウォッチポイント活動中信号及び前記ジャンプ&リンク命令実行完了信号の表明に応答して、ジャンプ&リンク命令評価有効化信号を生成する段階；

ー 前記ジャンプ&リンク命令評価有効化信号の受理（の受理）に応じて評価準備完了信号を生成する段階；及び

ー 前記評価準備完了信号及び前記評価真信号の受理に応じて命令誤予測信号を生成する段階、

を含んで成る方法。

169. 中央処理ユニットの中で、投機的に発行された予測された分岐命令及びジャンプ&リンク命令を含む予め定められた投機的発

行済み命令のための複数の誤予測信号を同時に生成するための方法において、

ー 前記投機的発行済み命令を監視するためにウォッチポイントが形成されつつあることを表示するウォッチポイント活動中信号を受理する段階；

ー 前記投機的発行済み命令の各々について前記CPU内に規定されたデータ記憶構造の中でウォッチポイント番号及び付随するウォッチポイントレジスタを割振りする段階；

ー ウォッチポイントがジャンプ&リンク命令について形成されつつあることを表示するウォッチポイントジャンプ&リンク有効化信号を受理する段階；

ー 前記1つの投機的発行済み命令に関係するウォッチポイントデータを受理し、前記割振られたウォッチポイントレジスタ内に前記ウォッチポイントデータを記憶する段階；

ー 各々の発行済み分岐命令について、分岐命令のための複数の誤予測信号を同時に生成する段階であって、

・条件コード選択論理回路から評価条件コード信号を受理する段階；

・ウォッチポイント記憶要素からウォッチポイント条件信号を受理する段階；

- ・ウォッチポイント記憶要素から条件コードタイプ制御信号を受理する段階；
- ・各実行結果ソースユニットから有効化制御信号を受理する段階；
- ・分岐のための条件が正しく予測されているか否かを評価するため各々の前記実行結果ソースユニットのための前記有効化制御信号の表明された状態を含め予め定められた状態コード評価規則に従っ

て前記ウォッチポイント条件信号と前記評価条件コード信号を比較する段階；

- ・前記分岐命令が正しく予測された場合には、分岐真信号を表明する段階；

を含む段階；及び

- ・前記分岐真信号が表明されたという決定に応答して評価真信号を生成する段階；そして

ー 各々の前記発行済みジャンプ&リンク命令について、複数のジャンプ&リンク誤予測信号を同時に生成する段階であって、

- ・プログラムカウンタデータを記憶するための複数の記憶場所をもつ標的アドレスデータ構造を規定する段階；

- ・前記ジャンプ&リンク命令が発行された時点で前記標的アドレスデータ構造記憶場所のうちの1つに代替的な次のプログラムカウンタを書込み、この記憶された代替的な次のプログラムカウンタを前記命令と結びつける段階；

- ・実行結果ソースユニット内での前記ジャンプ&リンクの実行中に計算上の次のプログラムカウンタ値を計算する段階；

- ・前記ジャンプ&リンク命令の実行の完了時点でジャンプ&リンク命令実行完了信号を生成する段階；

- ・前記実行結果ソースユニットから前記計算上の次のプログラムカウンタを受理する段階；

- ・前記記憶場所から前記ジャンプ&リンク命令に付随する前記記憶された代替的な次のカウンタを読取る段階；

- ・前記代替的な次のプログラムカウンタ値を前記計算上の次のプログラムカウンタ値と比較する段階；

- ・前記代替的な次のプログラムカウンタ値が前記計算上の次のプログラムカウ

ンタ値と一致する場合、ジャンプ&リンカー致信号を

生成する段階；及び

- ・前記ジャンプ&リンカー致信号と前記ウォッチポイントジャンプ&リンク有効化信号の両方の表明に応じてジャンプ&リンク真信号を生成する段階、

- ・前記ジャンプ&リンカー致信号が表明されたという決定に回答して評価真信号を生成する段階；

- ・前記ウォッチポイント活動中信号、前記命令ウォッチポイント番号信号、前記ウォッチポイント活動中信号及び前記ジャンプ&リンク命令実行完了信号の表明に回答して、ジャンプ&リンク命令評価有効化信号を生成する段階；

を含む段階

- ー 投機的発行済み分岐命令のための前記結果データの捕捉の結果としての少なくとも1つのデータ捕捉有効化信号の受理又は前記ジャンプ&リンク命令評価有効化信号の受理に回答して、評価準備完了信号を生成する段階；及び

- ー 前記評価準備完了信号及び前記評価真信号の両方の受理に回答して命令誤予測信号を生成する段階、

を含んで成る方法。

170. ー 各々の実行結果ソースユニットから有効化制御信号を受理する段階；

- ー 各々の実行結果ソースユニットから結果データタイプ信号を受理する段階；

- ー 前記結果データが前記実行結果ソースユニットのいずれから受理されることになるかを表示するセレクトソース信号及び、前記結果データが予測される現行クロックサイクルとの関係における時刻の表示を受理する段階；

- ー 予め定められた規則に従って前記有効化制御信号、前記結果デ

ータタイプ信号及び前記セレクトソース信号の受理に回答してソース選択信号を生成する段階；

- ー 前記実行結果ソースユニットから実行結果データ信号を監視する段階；

- ー 前記到着する実行結果データ及び前記ソース選択信号に応じて前記選択され

たソースから実行結果データを捕捉する段階；及び

－ 前記プロセッサ内のデータ記憶装置内に評価のため前記捕捉された実行結果データを記憶する段階；

をさらに含んで成り、

かくして予め定められた命令が待っている結果データは、複数の実行結果ソースユニットから同時に捕捉される請求項169に記載の方法。

171. 前記実行結果ソースユニットには命令実行ユニットとレジスタリネームユニットが含まれている請求項169に記載の方法。

172. － 前記予め定められた投機的発行済み命令が制御転送命令、分岐命令、ジャンプ&リンク命令及びそれらの組合せから成るグループの中から選択され；

－ 前記条件コードタイプ制御信号には、WP_XCC, WP_ICC, WP_FCC及びその組合せから成るグループの中から選択されたものが含まれており、

－ 前記分岐真信号がBR_TRUE 信号を含み；

－ 前記評価条件コード信号がEVAL_CC 信号を含んでおり；

－ 前記ウォッチポイント条件信号にはWP_COND 信号が含まれており；

－ 前記有効化信号がWP_JMPL 信号を含んでおり、前記第1の状態がWP_JMPL = 0であり、前記第2の状態がWP_JMPL = 1であり、

前記ソース選択信号が、SEL_BR_XCC, SEL_BR_ICC, SEL_FXU_XCC,

SEL_FXU_ICC, SEL_FXAGU_CC, SEL_FXAGU_ICC、及びSEL_FPU_FCC信号から成るグループの中から選ばれたものを含んでおり；

－ 前記ジャンプ&リンク命令実行完了信号には、FXAGU_JMPL信号が含まれており；

－ 前記ジャンプ&リンクチェックポイント番号がFXAGU_CHKPNT_DEC信号を含み；

－ 前記ジャンプ&リンク一致信号がJMPL_MATCH信号を含み；

－ 前記ジャンプ&リンク真信号がJMPL_TRUE 信号を含み；

－ 前記ウォッチポイントジャンプ&リンク信号がWP_JMPL 信号を含み；

- 前記評価真信号がEVAL_TRUE 信号を含み；
- 前記評価準備完了信号がEVAL_RUE 信号を含み；
- 前記評価準備完了信号がEVAL_READY信号を含み；
- 前記評価有効化信号がEVAL_ENABLE 信号を含み；
- 前記ウォッチポイント活動中信号がWP_ACTIVE_VEC 信号を含み；
- 前記ウォッチポイントジャンプ&リンク有効化信号がWP_JMPL 信号を含み；
- 前記命令誤予測信号がWP_MISPRED_VEC信号を含む

請求項169 に記載の方法。

173. 命令発行ユニット及び複数の命令実行ユニットを有する中央処理ユニットの中で、投機的発行済みの予測された分岐命令及びジャンプ&リンク命令を含む、予め定められた投機的発行済み命令のための複数の誤予測信号を同時に生成するための方法において、

- 命令シーケンスの実行の結果としてもたらされる条件を予測する段階；
- 前記実行シーケンスの実行を完了する前に前記予測された条件

に基づいて制御転送命令を投機的に発行する段階；

- 前記CPU のデータ構造内でウォッチポイント記憶レジスタを割振りし、前記投機的に発行された命令のうちの特定の1つの命令と前記割振られたレジスタを結びつける段階；
- 前記制御転送命令のための正しい方向を有する評価された条件との後の比較のために、前記割振られたウォッチポイントレジスタ内に前記予測された条件を記憶する段階；及び

各々の投機的に発行された命令について、

- 前記実行ユニットによって生成された実行結果データを監視する段階、
- 各々の前記ウォッチポイントレジスタの中に記憶された前記実行前予測情報を前記実行ユニットから受理した実行後評価された条件情報と比較して、前記予測された条件情報が前記命令シーケンスについて前記評価された条件情報に一致しているか否かを見極める段階；及び
- 前記命令のための前記予測情報が前記命令のための前記既知の情報と一致し

ない場合に、誤予測された条件情報の結果として誤って発行された各々の投機的発行済み命令についての誤予測表示を生成する段階、
を含んで成る方法。

174. 命令発行ユニット、命令実行ユニット、及びデータを記憶するためのメモリ記憶ユニットを有する中央処理ユニットの中で、このユニット内の単一の共用記憶ユニット内の共通のデータ構造の中に代替的分岐アドレス及び予測されたジャンプ&リンクアドレスを並行して記憶するための方法において、

ー 発行された各々の制御転送命令についてそれが発行された時点で発行ユニットからの制御転送命令発行済み信号を生成する段階で

あって、前記制御転送命令発行済み信号には、分岐命令又はジャンプ&リンク命令として制御転送命令を識別するべくウォッチポイントタイプの信号が含まれている段階；

ー 予測された分岐及びジャンプ&リンク命令の命令復号中に代替的な次のプログラムカウンタを決定する段階；

ー 代替的分岐アドレスを記憶するための前記記憶ユニット内のレジスタ場所を識別するウォッチポイント書込み有効化制御信号及びウォッチポイント書込みアドレス信号を生成する段階であって、このウォッチポイント書込み有効化制御信号が、チェックポイント又はウォッチポイントを作る時に表明される、段階；

ー 前記ウォッチポイント書込みアドレスにより表示されたレジスタ場所で前記記憶ユニット内に前記ウォッチポイント代替次プログラムカウンタを書込む段階；

を含み、ここで

ー 前記ウォッチポイント代替次プログラムカウンタには、前記制御転送命令がジャンプ&リンク命令である場合予測された標的フェッチプログラムカウンタが含まれ、前記ウォッチポイント代替次プログラムカウンタには、前記制御転送命令が予測された分岐命令である場合に代替的分岐方向のためのフェッチプログラムカウンタが含まれ、さらに、

ー 前記ウォッチポイント命令タイプ信号に基づいて別々のアドレス可能な記憶

場所の中に前記ジャンプ&リンク命令又は前記分岐命令の各々のための前記代替次プログラムカウンタを記憶する段階、
を含んで成る方法。

175. 前記制御転送命令が、分岐命令、ジャンプ&リンク命令及びそれらの組合せから成るグループの中から選択され、

— 前記制御転送命令発行済み信号には、ジャンプ&リンク信号成

分WP_JMPL を含むDO_WTCHPNT信号が含まれており；

— 前記代替次プログラムカウンタがWP_ANPC 信号を含み；

— 前記チェックポイント書込み有効化信号がMAKE_CHKPNT 信号を含み；

— 前記チェックポイント書込みアドレス信号がNEXT_CHKPNT 信号を含み；

— 前記データ構造中の前記記憶場所がTARGET_RAM内に位置づけられ

— 前記分岐命令のための情報項目にはさらに、XCC により左右される分岐、IC
C により左右される分岐、FCC により左右される分岐又はジャンプリンク命令と
して命令を指定する分岐タイプフィールド；CONDフィールド；及び条件コードタ
グフィールドが含まれている、

請求項174 に記載の方法。

176. — 各々の実行結果ソースユニットから有効化制御信号を受理する段階
；

— 各々の実行結果ソースユニットから結果データタイプ信号を受理する段階；

— 前記実行結果ソースユニットのいずれから前記結果データが受理されること
になるかを表示するセレクトソース信号及び前記結果データが予想される現行ク
ロックサイクルとの関係における時刻の表示を受理する段階；

— 予め定められた規則に従って前記有効化制御信号、前記結果データタイプ信
号及び前記セレクトソース信号の受理に応答してソース選択信号を生成する段階
；

— 前記実行結果ソースユニットから実行結果データ信号を監視する段階；

— 前記到着する実行結果データ及び前記ソース選択信号に応じて前選択された

ソースから実行結果データを捕捉する段階；及び

ー 前記プロセッサ内のデータ記憶装置内に評価のため前記捕捉された実行結果データを記憶する段階；

をさらに含んで成り

かくして予め定められた命令が待っている結果データは、複数の実行結果ソースユニットから同時に捕捉される請求項174に記載の方法。

177. 投機的に発行された命令の誤予測が発生した後、命令再フェッチのための適正な再フェッチ命令アドレスを送る段階をさらに含んで成り、この送信段階には、

ー 前記命令の実行に先立ち各々の投機的に発行された予測された命令について、この予測された命令のための代替的执行経路を指定する代替的経路プログラム命令アドレス及び、前記CPU内のメモリユニット内に規定された標的アドレス構造内のウォッチポイント要素の中で前記予測された実行を左右する予測情報を、同時に記憶する段階；

ー 各々の前記予測された命令の実行の完了時点で、中に実行前代替経路及び予測情報が記憶されている前記命令に対応するウォッチポイント要素番号及び前記記憶された条件値及び代替的プログラム命令アドレスとの比較のための計算上の適正な条件値を識別する、命令実行完了状況を表示する信号を同時に生成する段階；

ー 前記記憶された実行前代替経路及び予測情報と各単一命令のための前記生成された計算上の条件データ値とを比較し、予測上の条件データがこれらの命令のいずれかについての前記計算上の適正な条件データ値と一致する場合には一致信号を生成すること及び、分岐命令又はジャンプ&リンク命令が正しく予測されたか又は予測が

正しくないかを評価し各々の誤予測命令について誤予測信号を生成することを含めて、各単一命令についての予測を同時に評価する段階；

ー 前記誤予測からの前記CPUの回復を優先順位づけし、予め定められた優先順位規則に基づいて特定の高優先順の誤予測を選択する段階；

- ー 前記特定の高優先順誤予測命令に付随するチェックポイントを識別し、前記特定の誤予測命令から前記プロセッサを回復するためのCPU バックアップ信号を生成する段階；
- ー 前記標的アドレスデータ構造から前記特定の誤予測命令のための代替次プログラムカウンタを読取る段階；及び
- ー 前記再フェッチされた命令に基づき前記CPU の実行を再開させるため発行ユニットに対して前記読取り代替次プログラムアドレスを送る段階、
が含まれている請求項174 に記載の方法。

178. 命令フェッチコンポーネントを含む命令発行ユニット、命令実行ユニット、データ記憶ユニットをもつ中央処理ユニットの中で、投機的に発行された命令の誤予測が発生した後命令フェッチのために適正なフェッチアドレスを送るための方法において

- ー 前記命令の実行に先立ち各々の投機的に発行された予測された命令について、この予測された命令のための代替的実行経路を指定する代替的経路プログラム命令アドレス及び、前記CPU 内のメモリユニット内に規定された標的アドレス構造内のウォッチポイント要素の中で前記予測された実行を左右する予測情報を、同時に記憶する段階；
- ー 各々の前記予測された命令の実行の完了時点で、中に実行前代替経路情報及び予測情報が記憶されている前記命令に対応するウォ

ッチポイント要素番号及び前記記憶された条件値及び代替的プログラム命令アドレスとの比較のための計算上の適正な条件値を識別する、命令実行完了状況を表示する信号を同時に生成する段階；

- ー 前記記憶された実行前代替経路及び予測情報と前記投機的発行済み命令のための前記生成された計算上の条件データ値とを比較し、予測上の条件データがこれらの命令のいずれかについての前記計算上の適正な条件データ値と一致する場合には一致信号を生成すること及び、分岐命令又はジャンプ&リンク命令が正しく予測されたか又は予測が正しくないかを評価し各々の誤予測命令について誤予測信号を生成することを含めて、前記投機的発行済み命令についての前記記憶さ

れた予測情報を同時に評価する段階；

- ー 全ての誤予測された命令の中から1つの誤予測された命令を選択する段階；
 - ー 前記選択された誤予測命令に付随するウォッチポイントを識別する段階；
 - ー 前記データ構造内に前記選択されたウォッチポイント番号を指標づけすることによって前記標的アドレスデータ構造から前記選択された誤予測命令のための代替的プログラムカウンタアドレスを読取る段階；及び
 - ー 望ましい命令シーケンス経路に沿ってCPU 命令を開始するべく命令再フェッチのために前記命令発行ユニットに対し前記読取られた代替次プログラムカウンタアドレスを送る段階、
- を含んで成る方法。

179. 前記1つの誤予測された命令を選択する段階には、前記CPU 内で発生する誤予測を優先順位づけし予め定められた優先順位規則に基づいて高優先順位の誤予測を選択する段階が含まれている請求項178 に記載の方法。

180. 前記投機的に発行された命令は、予測された制御転送命令、予測された分岐命令、ジャンプ&リンク命令及びそれらの組合せから成るグループの中から選択される、請求項178 に記載の方法。

181. 前記同時記憶、同時生成及び同時評価の段階には、それぞれ、同じCPU クロックサイクル内での記憶、生成及び評価段階が含まれている、請求項178 に記載の方法。

182. 命令フェッチコンポーネントを含む命令発行ユニット、命令実行ユニット、データ記憶ユニットをもつ中央処理ユニットの中で、投機的に発行された命令の誤予測が発生した後命令フェッチのために適正なフェッチアドレスを送るための方法において

- ー 予測された分岐命令の発行に応じて、前記CPU 内のメモリユニットのデータ構造内に前記分岐命令に付随するウォッチポイント要素内のウォッチポイント代替プログラムカウンタデータを記憶する段階であって、ここで前記ウォッチポイント代替プログラムカウンタデータは、予測がとられた場合には分岐の次のアドレスであり、予測がとられない場合には分岐一標的アドレスであるような段階；

- ー 予測されたジャンプ&リンク命令の発行に応じて、前記CPU内のメモリユニットのデータ構造内に前記ジャンプ&リンク命令に付随するウォッチポイント要素の中の予測されたジャンプ&リンク標的アドレスを記憶する段階；
- ー 前記発行された命令にウォッチポイント要素を割振るべく各々の分岐及びジャンプ&リンク命令のためのウォッチポイント有効信号が発行されることを表明する段階；
- ー 前記ウォッチポイント有効信号の表明に応じて、そのウォッチポイント活動中ベクトル信号内で識別されたチェックポイント番号を使用する予測された命令が発行されまだ解決されていないことを表示するべくこのウォッチポイント活動中ベクトル信号を表明する

段階；

- ー 各々のジャンプ&リンク命令について、適正なジャンプ&リンク標的アドレスを計算すること、アドレス生成ユニットがジャンプ&リンク命令の実行を終えた時点でアドレス生成ユニットジャンプ&リンク命令完了信号を表明すること、ジャンプ&リンク命令のチェックポイント番号を指定するアドレス生成ユニットジャンプ&リンクチェックポイント識別信号を生成すること、及び計算された適正なジャンプ&リンク命令標的アドレスを指定するアドレス生成ユニットジャンプ&リンクデータ信号を生成することを含めた、前記ジャンプ&リンクの実行を完了する段階；
- ー 各々の予測された分岐命令について、分岐方向を左右する条件コード値を計算すること及び前記評価準備完了信号の受信に応答して、ウォッチポイント番号に対応する評価条件コード信号が利用可能になった時点でこの評価条件コード信号を捕捉することを含む、前記分岐命令の実行を完了する段階；
- ー ・*前記アドレス生成ユニットジャンプ&リンクチェックポイント識別信号によって標識づけされた前記標的アドレスメモリユニット内の記憶場所の中の予測されたジャンプ&リンクアドレスを読取ること；

＊前記予測されたジャンプ&リンクアドレスを前記計算されたジャンプ&リンクアドレスと比較して比較結果を生成すること；

* 予測されたアドレス及び計算された適正なアドレスが同じである場合にジャンプ&リンク一致信号を生成すること；

* 前記ウォッチポイント要素内に記憶された前記ウォッチポイントジャンプ&リンク有効化信号と前記ジャンプ&リンク一致信号を比較すること；及び

* 前記信号が一致し、前記ウォッチポイントがジャンプ&リン

ク命令について形成されジャンプ&リンク命令が適正に予測されたことを表わしている場合、ジャンプ&リンク真出力信号を生成することによって、ジャンプ&リンク命令が適切に予測されたか否かを決定する段階；

・ * 条件コードセレクト論理ユニットからの条件コード評価信号を受理すること；

* 命令に付随するウォッチポイント記憶要素内に記憶されたウォッチポイント条件信号を含むウォッチポイントレジスタ内に記憶されたウォッチポイントを読取ること；

* 分岐を発行するときに作成された予測が適正であるか否かを評価するべく予め定められた規則に基づき前記分岐命令に対応するウォッチポイント要素内に記憶された前記ウォッチポイント条件データと前記評価条件コード信号を比較すること；及び

* 前記条件コード評価信号及び前記ウォッチ条件信号である場合、前記特定のウォッチポイントのための分岐真信号を生成することにより、分岐命令が適正に予測されたか否かを決定する段階、及び

・ 前記分岐真信号とジャンプ&リンク真信号の論理和をとり、前記分岐真又はジャンプ&リンク信号が真として表明された場合に評価真信号を生成する段階；を含み、分岐命令又はジャンプ&リンク命令のいずれか（ただしその両方ではない）に対する前記ウォッチポイント割振りが、同じウォッチポイント要素のための前記分岐真信号及び前記ジャンプ&リンク信号の並行表明を制限している、評価論理ユニット内の各々の単一命令のための予測を評価する段階；

ー 前記ジャンプ&リンク命令完了信号がCPU サイクル中に各命令について表明された後評価準備完了信号を生成する段階；

ー 前記評価真信号のいずれかが表明されて、誤信号が検出されたことを表わした場合、複数の評価真論理ユニットの各々から生成された前記評価真信号からの誤予測を検出する命令ウォッチポイント誤予測信号を生成し、前記CPU内の精確な状態を維持するために前記誤予測信号を精確な状態のユニットに伝達する段階；

ー 生成された全てのウォッチポイント誤予測済み信号を優先順位づけし、予め定められた優先順位づけ規則に従ってCPU回復のための前記誤予測の1つを識別する段階；

ー 精確な状態のユニットの中で、誤予測された分岐命令又は誤予測されたジャンプ&リンク命令のいずれかによる誤予測から回復する目的でCPUがバックアップされることになるチェックポイント実行済み命令を識別するバックアップチェックポイント信号を生成する段階；

ー 精確な状態のユニットの中で、バックアップが誤予測されたジャンプ&リンク命令のせいである場合にのみジャンプ&リンク誤予測信号を生成する段階；

ー 前記識別された予測、誤予測のタイプ及びウォッチポイント／チェックポイント番号を含む因子に基づいて前記メモリ記憶装置内の1つの場所を選定する段階であって；

・ウォッチポイント誤予測信号の受理に応じて前記精確な状態のユニット内でプロセッサバックアップ開始信号を生成する段階；

・誤予測がCPU内で発生したこと及びCPUが、1つの代替プログラムカウンタ値へとプログラムカウンタをリセットすることを含むより早期の状態までバックアップされるべきであることを表わすプロセッサバックアップ開始信号を受理する段階；

・前記メモリ記憶領域内へチェックポイント／ウォッチポイント番号を識別する指標を提供するバックアップチェックポイント信号

を受理し、前記指標を用いて命令フェッチのための適正な分岐アドレス又は適正なジャンプ&リンクアドレスを記憶する場所の内容を読取る段階；

・前記メモリ記憶領域内へウォッチポイント及びチェックポイント番号の形で

指標を提供するアドレス生成ユニットチェックポイント信号を受理し、この指標を用いて代替次プログラムカウンタを記憶する場所の内容を読取る段階；

- ・誤予測の検出に応じてCPU バックアップが要求された時点で、前記精確な状態のユニットからバックアップ作成制御信号を受理する段階；

- ・アドレス生成ユニットがジャンプ&リンク命令の実行を終えた時点でアドレス生成ユニットジャンプ&リンク命令完了信号を受理する段階；

- ・前記メモリ記憶領域内の特定のウォッチポイント要素を特定するアドレス読取り指標として前記精確な状態ユニット又は前記アドレス生成ユニットチェックポイント入力信号から前記バックアップチェックポイントの1つを選択するべくマルチプレクサに対して制御セクタ入力信号として前記ジャンプ&リンク完了信号及びバックアップ作成信号を印加する段階；

- ・誤予測された命令が、バックアップ作成及びジャンプ&リンク制御信号によって表示される通りにジャンプ&リンク命令の分岐命令であるか否かに基づいて、読取るべきメモリ記憶場所アドレスを選択する段階；を含む段階；及び

- ー 前記誤予測から回復するべく新しい命令をフェッチするため代替次プログラムカウンタを選択する段階であって、

- ・前記誤予測がジャンプ&リンク命令誤予測である場合にジャンプ&リンク誤予測信号を生成する段階；

- ・マルチプレクサ選択制御信号、複数のマルチプレクサ入力信号及び前記制御信号によって選択された入力信号のための1つのマルチプレクサ出力をもつマルチプレクサに対して前記誤予測信号を印加する段階；

- ・前記マルチプレクサ選択制御信号が、誤予測が誤予測された分岐命令であることを表わす第1の状態を有する場合に、マルチプレクサ出力信号として前記データ記憶装置内の前記データ構造内の前記記憶場所の中に記憶された代替次プログラムカウンタを含む第1のマルチプレクサ入力信号を選択する段階；

- ・前記マルチプレクサ信号が、誤予測が発生したこと及び誤予測が誤予測されたジャンプ&リンク命令であることを表わす第2の状態を有する場合に、マルチプレクサ出力信号として計算上のジャンプ&リンクアドレスを含む第2・マルチ

プレクサ入力信号を選択する段階；を含む段階；

ー ・ジャンプ&リンク命令のための予測された標的フェッチプログラムカウンタ又は予測された分岐命令のための代替分岐方向についてのフェッチプログラムカウンタを含む読取り代替プログラムカウンタアドレス信号を生成する段階；及び

・フェッチユニットが適正なアドレスから命令を再フェッチすることができるように命令フェッチユニットに対して前記選択されたマルチプレクサ出力の中の前記読取られた代替プログラムアドレス信号を送る段階；

を含む、前記CPU の実行を再開する段階；

を含んで成る方法。

183. ー 前記再フェッチアドレスが、分岐命令アドレス及びジャンプ&リンク命令アドレスから成るグループの中から選択され；

ー 前記ジャンプ&リンク命令完了信号がFXAGU_JMPL信号を含み；

ー 前記チェックポイント識別信号がFXAGU_CHKPNT信号を含み；

ー 前記アドレス生成ユニットデータ信号がFXAGU_DATA信号を含み；

ー 前記ジャンプ&リンケ一致信号がJMPL_MATCH信号を含み；

ー 前記ウォッチポイントジャンプ&リンク有効化信号がWP_JMPL 信号を含み；

ー 前記ジャンプ&リンク真（信号）がJMPL_TRUE 信号を含み；

ー 前記条件コード評価信号がEVAL_CC 信号を含み；

ー 前記ウォッチポイントデータが、WP_COND, WP_XCC, WP_ICC 、及びWP_FCCデータ信号を含み；

ー 前記ウォッチポイント条件信号がWP_COND 信号を含み；

ー 前記分岐真信号がBR_TRUE 信号を含み；

ー 前記評価真信号がEVAL_TRUE 信号を含み；

ー 前記評価準備完了信号がEVAL_READY信号を含み；

ー 前記ウォッチポイント活動中信号がWP_ACTIVE_VEC 信号を含み；

ー 前記ウォッチポイント誤予測信号がWP_MISPRED_VEC信号を含み；

ー 前記バックアップチェックポイント信号がBACKUP_CHKPNT 信号を含み；

- ー 前記ジャンプ&リンク誤予測信号がMIS_PRED信号を含み；
- ー 前記プロセッサバックアップ開始信号がDO_BACKUP 信号を含み；
- ー 前記バックアップチェックポイントがBACKUP_CHKPNT 信号を含み；
- ー 前記バックアップ実施制御信号がMAKE_BACKUP 信号を含み；
- ー 前記生成ユニットジャンプ&リンク命令完了信号がFXAGU_JMPL信号を含み；
- ー 前記選択されたバックアップチェックポイント信号がBACKUP_CHKPNT 信号を含み；
- ー 前記アドレス生成（前記）ユニットチェックポイントがFXAGU_CHKPNT信号を含み；
- ー 前記ジャンプ&リンク誤予測信号がMISPRED 信号を含み；
- ー 前記代替次プログラムカウンタがWR_ANPC 信号を含み；
- ー 前記計算上のジャンプ&リンクアドレスがFXAGU_DATA信号を含み；
- ー 前記代替プログラムカウンタアドレスがRD_ANPC 信号を含み；
- ー 前記ジャンプ&リンク一致信号を前記ウォッチポイントジャンプ&リンク有効化信号が、前記信号のブール論理積をとる段階を含んでいる、請求項182 に記載の方法。

184. バックアップポイントでの命令の後に逐次的に発生する命令のためのチェックポイントをキルするべく前記誤予測からの回復中に誤予測の識別に応じて命令キル信号を表明する段階；

- ー 前記キル信号の表明に応じて、ウォッチポイント活動中信号及びウォッチポイント誤予測信号の生成を抑制することにより各々のキルされた命令に対応するウォッチポイント要素を非活動化する段階
- をさらに含んで成る請求項182 に記載の方法。

185. 前記ウォッチポイント活動中ベクトル信号及びウォッチポイント誤予測信号は、各ウォッチポイントについてのウォッチポイント活動中情報を同時に記憶し同じ信号内で伝送できるように前記CPU 内の割振り可能なチェックポイントの数に対応するベクトルビット位置の数を各々有するマルチホットベクトルとし

て各々別々に

コード化されている請求項182 に記載の方法。

186. 比較時間を短縮するようにデータ順方向送りバスタグと条件コードデータを比較するための方法において、

ー ・ 現行の発行ウインドウ信号内の現行のリネーム済み条件コードを各々のデータ順方向送りバスタグに比較し；

・ 先行するサイクルの条件コードタグを各々のデータ順方向送りバスタグと比較すること、

によって、先行するサイクル内の条件コードタグと現行の発行ウインドウサイクル内の現在リネームされている条件コードタグの間でデータ順方向送りタグの比較を分離する段階を含み、

ー 先行するサイクル中の前記条件コードタグ及び現行の発行ウインドウ内の現在リネームされている条件コードタグとの前記データ順方向送りバスタグの前記比較は、単一の非分割作業で行なわれた場合に比べさらに迅速に行なわれるような方法。

187. 現在リネームされている条件コードの前記比較を行なう前にラッチ回路内で現行の発行ウインドウ信号中の現在リネームされている条件コードタグをラッチングする段階をさらに含んで成り、

各々のデータ順方向送りバスタグに対し現行送信ウインドウサイクル内の現在リネームされている条件コードタグを比較する前記段階には、各々のデータ順方向送りバスタグに対する前記ラッチされた信号の比較が含まれ、かくしてタイミングクリティカリティが削除される請求項186 に記載の方法。

188. 命令発行ユニット、条件コードレジスタリネームユニットを含む論理及び物理レジスタをリネームするためのレジスタリネームユニット、複数の実行ユニット及び、前記実行ユニットの各々が中央処理ユニットのその他のコンポーネントまで実行結果データを伝達するためのデータ順方向送りバスを有する中央処理ユニットに

において、前記複数の実行ユニットの各々からの実行結果データを同時に捕捉するための方法において、

ー 各々の前記実行ユニットについて、先行するCPU クロックサイクル内でリネームされた条件コードレジスタを識別するものの現行サイクルでリネームされた条件コードレジスタを識別しない前記条件コードレジスタリネームユニットから受理した早期条件コードレジスタタグ信号を、現行のCPU クロックサイクル内で前記実行ユニットから到着した条件コードレジスタタグ信号成分を含む有効な実行結果データ信号と比較し、前記比較動作が、前記条件コードレジスタタグ信号と前記条件コードレジスタタグ信号成分を含む前記有効な実行結果データ信号の間での一致を識別した場合に、各々の前記実行ユニットについての条件コードタグ現行一致信号を生成する段階；及び

ー 各々の前記実行ユニットについて、現行サイクル内でリネームされた条件コードレジスタを識別することを含め先行するCPU クロックサイクル内でリネームされた条件コードレジスタを各々識別する先行サイクル中に前記条件コードレジスタリネームユニットから受理した複数の晚期条件コードレジスタタグ信号を、現行のCPU クロックサイクル内で前記実行ユニットから到着した条件コードレジスタタグ信号成分を含む有効な実行結果データ信号と比較し、前記比較動作が、前記条件コードレジスタタグ信号の1つと前記複数の条件コードレジスタタグ信号成分の1つの間での一致を識別した場合に、前記複数の条件コードレジスタタグ信号のうちのいずれが一致したかを識別する各々の前記実行ユニットについての現行条件コードタグのアレイー晚期ー一致信号を生成する段階、を含んで成る方法。

189. ー 前記早期条件コードレジスタタグ信号がCC_TAG_C信号

を含み；

- ー 前記条件コードレジスタリネームユニットがFSR/CCRFRNユニットを含み；
- ー 前記有効な実行結果データ信号は、FXU_XICC_TAG_F信号、FXAGU_XICC_TAG_F信号及びFPU_FCC_TAG_F信号から成るグループの中から選択されており；
- ー 前記実行ユニットは、固定小数点実行ユニット、固定小数点／アドレス生成

ユニット及び浮動小数点ユニットから成るグループの中から選択され

- ー 前記条件コードタグ現行一致信号は、FXU_CURR_MATCH信号、FXAGU_CURR_MATCH信号及びFPU_CURR_MATCH信号から成るグループの中から選択され；
- ー 前記複数の晚期条件コードレジスタタグ信号が、複数のCC_TAG_RENAMED信号を含んでおり；
- ー 前記条件コードレジスタリネームユニットがFSR/CCRFRNユニットを含み；
- ー 前記現行の条件コードタグアレイ晚期一致信号が、各々の前記実行ユニットについて、FXU_ARRAY_LATE_MATCH信号、FXAGU_ARRAY_LATE_MATCH信号及びFPU_ARRAY_LATE_MATCH信号から成るグループの中から選択される請求項188に記載の方法。

190. ー 前記晚期条件コードレジスタタグを記憶するため前記CPUの中で複数のアドレス可能なタグ記憶要素を含む晚期条件コードタグアレイデータ記憶装置を提供する段階；

- ー 前記条件コードレジスタリネームユニットから前記晚期条件コードレジスタタグ信号成分を受理する段階；
- ー 前記条件コードレジスタリネームユニットから条件コードリネーム済み信号を受理する段階；

ー 前記命令発行ユニットからのウォッチポイント実施信号から誘導されたウォッチポイント番号識別成分信号を含む予測実施信号を受理する段階；

ー 前記条件コードリネーム済み信号及び前記予測実施信号が両方共表明された場合に、前記タグアレイデータ記憶装置内への前記条件コードレジスタタグ信号の各々の書込みのアドレスを制御するためのタグアレイ書込み有効化信号を生成する段階；

ー 前記タグアレイ書込み有効化信号が表明された場合に前記ウォッチポイント識別番号により識別された前記アドレス可能なタグ記憶要素のうちの1つの中に前記受理した晚期条件コードレジスタタグ信号成分を記憶する段階；及び

ー 前記比較動作において使用するため各々の前記実行ユニットについて各々の前記タグ記憶要素から各々の前記記憶された条件コードレジスタタグを読取る段

階、

を含んで成る請求項188 に記載の方法。

191. 前記条件コードリネーム済み信号が、CC_RENAMED信号を含み；

－ 前記予測実施信号がDO_PREDICTVEC信号を含み、ウォッチポイント実施信号がDO_WTCHPT 信号を含み；

－ 前記タグアレイ書込み有効化信号がARRAY_WRITE_VEC 信号を含んでいる、請求項190 に記載の方法。

192. 中央処理ユニット内で複数の命令の実行結果を同時に監視するための装置において、

－ 命令を発行するための発行ユニット；

－ 命令を実行するための実行ユニット；

－ 前記中央処理ユニット内で前記実行ユニットからその他のユニ

ットまで実行結果を伝達するデータ順方向分配バス；

－ プロセッサ内で前記データ順方向分配の上で前記実行ユニットからの実行結果信号を受理するべく結合されたウォッチポイントデータを記憶するため複数のウォッチポイントレジスタを有するウォッチポイントユニット；

－ 前記命令が発行された時点で各々の投機的発行済みの予測された命令についてウォッチポイントデータを記憶するためのウォッチポイントレジスタを割振るための手段；

－ 前記ウォッチポイントユニット内で前記データ順方向送りバス上で伝達された実行結果信号を監視するための手段；

－ 記実行結果信号及び予め定められた規則に基づいて予め定められた事象の発生を検出するための手段；

－ 予測された投機的発行済み命令について予め定められた事象が検出された時点で、その他のCPU ユニットにそれを知らせるための手段；

－ 特定の命令について前記ウォッチポイントレジスタ内に記憶された前記ウォッチポイントデータとこの特定の命令に関する前記データ順方向送りバスの上で到着した前記結果信号と比較して、この特定の命令が投機的に発行された条件が

正しく予測されたか否かを評価するための手段；

－ 前記予測が適正でなかったことを前記評価が表示した場合に、前記ウォッチポイント要素を割振り解除し実行を続行するための割振り解除手段；

－ 前記予測が誤っていたということを前記評価が表示した場合に誤予測を補正するべく既知の条件に対応する命令を再フェッチ発行できるように、前記発行ユニットに対し新しいアドレスを供給するための手段；

－ 前記誤予測を取消することができるように、前記中央処理ユニットをより早期の状態までバックアップするための手段；

－ 前記再フェッチされた命令に基づいて適正な命令ストリーム内の命令について実行を開始するための手段；及び

－ 前記実行が完了した時点で前記ウォッチポイント要素を割振り解除するための手段

を含んで成る装置。

193. 中央処理ユニット（CPU）内で複数の実行結果生成ユニットから予め定められた命令が待っている結果データを同時に捕捉するための装置において、

－ 各々の結果データソースから有効化制御信号を受理するための手段；

－ 各々の結果データソースから結果データタイプ信号を受理するための手段；

－ 前記実行結果生成ユニットのうちのいずれから前記結果データが受理されることになるかを表示するセレクトソース信号及び前記結果データが予想されている現行クロックサイクルとの関係における時刻の表示を受理するための手段；

－ 予め定められた規則に従って前記有効化制御信号、前記結果データタイプ信号及び前記セレクトソース信号に応じてソース選択信号を生成するための手段；

－ 前記実行結果生成ユニットから実行結果データ信号を監視するための手段；

－ 前記到着した実行結果データ及び前記ソース選択信号に応じて前記選択されたソースから実行結果データを捕捉するための手段；

及び

－ 前記プロセッサ内のデータ記憶装置内での評価のため前記捕捉

された実行結果データを記憶するための手段、
を含んで成る装置。

194. 前記捕捉用手段には、

- ー 各々有効化制御信号入力ポート、結果データタイプ信号入力ポート、及びセレクトソース信号入力ポートを内含する第1の複数の論理AND ゲート；
- ー 同じ結果データタイプ依存性をもつ前記複数のAND ゲートからの選択された出力と組合せるため、前記第1の複数の論理AND ゲートのうちの選択されたものに結合された第2の複数の論理ORゲート；
- ー 前記第1の複数の論理AND ゲート及び前記論理ORゲートのうちの選択されたゲートに結合された結果データソース入力ポートとセレクト条件コードタイプ入力ポートを各々有する第3の複数の論理AND ゲート；
- ー 各々前記第3の複数の論理ORゲートの出力を受理するための複数の入力ポートを有する多入力論理ORゲート；
- ー 条件コードデータを含む捕捉された実行結果データを含む前記多入力ORゲートの出力信号をラッチングするためのラッチ回路、
が含まれている請求項193 に記載の装置。

195. 中央処理ユニットの中で分岐命令及びジャンプ&リンク命令を含む予め定められた命令タイプについての複数の誤予測信号を同時に生成するための装置において、

- ー 分岐命令を監視するためにウォッチポイントが形成されつつあることを表示する第1の状態又はジャンプ&リンク命令のためのウォッチポイントが形成されつつあることを表示する第2の状態を有するウォッチポイントジャンプ&リンク有効化信号を受理するための手段；
- ー ・条件コードセレクト論理回路からの評価条件コード信号を受理するための手段；
 - ・ウォッチポイント記憶要素からのウォッチポイント条件信号を受理するための手段；
 - ・ウォッチポイント記憶要素からの条件コードタイプ制御信号を受理するため

の手段；

- ・各々の実行結果生成ユニットからの有効化信号を受理するための手段；

- ・分岐のための条件が正しく予測されたか否かを評価するため各々の前記実行結果生成ユニットについて前記有効化制御信号の表明された状態を含む予め定められた条件コード評価規則に従って前記ウォッチポイント条件信号と前記評価条件コード信号を比較するための手段；

- ・前記分岐が正しく予測されていた場合分岐真信号を生成するための手段；を含む、複数の発行済み分岐命令について誤予測信号を同時に生成するための手段；及び

- ・プログラムカウンタデータを記憶するための複数の記憶場所を有する標的データ構造を規定するための手段；

- ・ジャンプ&リンク命令が発行された時点で前記記憶場所のうちの1つに代替次プログラムカウンタを書込み、前記記憶された代替次プログラムカウンタを前記命令と結びつけるための手段；

- ・実行結果生成ユニット内で計算上の次プログラムカウンタ値を計算するための手段；

- ・前記実行結果生成ユニットから前記計算上の次プログラムカウンタを受理するための手段；

- ・前記記憶場所からの前記命令に付随する前記記憶された代替次

プログラムカウンタを読取るための手段；

- ・予め定められたプログラムカウンタ比較規則に従って前記計算上の次プログラムカウンタと前記代替次プログラムカウンタを比較するための手段；

- ・前記比較が真として評価した場合、ジャンプ&リンク一致信号を生成するための手段及び

- ・前記ジャンプ&リンク一致信号と前記ウォッチポイントジャンプ&リンク信号の両方の真の状態に応じて、ジャンプ&リンク真信号を生成するための手段、を含む、複数の発行済みジャンプ&リンク命令の各々についてジャンプ&リンク誤予測信号を同時に生成するための手段；

- ー 前記ジャンプ&リンク一致信号が表明されるか又は前記分岐真信号が表明されるかのいずれかに応えて、評価真信号を生成するための手段；
 - ー 少なくとも1つのソース選択信号の受理又は評価有効化信号の受理に応じて評価準備完了信号を生成するための手段であって、この評価有効化信号はウォッチポイント活動中信号、ウォッチポイントジャンプ&リンク有効化信号、ジャンプ&リンク命令実行完了信号及びジャンプ&リンク命令チェックポイント番号信号の表明に応じて生成されるような手段；及び
 - ー 前記評価準備完了信号及び前記評価真信号の受理に応じて命令誤予測信号を生成するための手段
- を含んで成る装置。

196. 中央処理ユニット内の単一の共用記憶ユニット内の共通のデータ構造の中に代替分岐アドレス及び予測されたジャンプ&リンクアドレスを同時に記憶するための装置において、

- ー 各々の発行された制御転送命令についてそれが発行された時点

で発行ユニットから制御転送命令発行済み信号を生成するための手段；

- ー 分岐命令又はジャンプ&リンク命令として命令を識別するべく前記制御転送命令発行済み信号内でウォッチポイントジャンプ&リンク信号を提供するための手段；
- ー プログラム制御論理ユニットにより計算された標的フェッチプログラムカウンタを特定する代替次プログラムカウンタを計算するための手段；
- ー 前記データ構造内の記憶場所の中に代替分岐アドレスを記憶するためのチェックポイント書込み有効化制御信号及びチェックポイントアドレスを生成するための手段であって、この書込み有効化制御信号がチェックポイントを作成するときに表明されるような手段；
- ー チェックポイント書込みアドレスにおいて記憶ユニット内にウォッチポイント代替次プログラムカウンタを書込むための手段；
- ー 予測された分岐及びジャンプ&リンク命令の命令復号の間に代替次プログラムカウンタを決定するための手段であって、ジャンプ&リンク命令のためにウォ

タッチポイントユニットに対して送られた前記ウォッチポイント代替次プログラムカウンタには、予測された標的フェッチプログラムカウンタが含まれ、予測された分岐命令のためにウォッチポイントユニットに対し送られた前記ウォッチポイント代替次プログラムカウンタには、代替分岐方向のためのフェッチプログラムカウンタが含まれているような手段；

— 発行された命令が分岐命令であるかジャンプ&リンク命令であるかを決定することを含め、フィールド選択及び書込み論理ユニット内で前記制御転送命令発行済み信号を復号して前記信号内の選択された情報フィールドの内容を決定するための手段であって、前記

JMPLフィールド内の第1の予め定められた状態は、命令がジャンプ&リンク命令であることを表示し、前記JMPLフィールド内の第2の状態は前記命令が分岐命令であることを表示しているような手段；及び

— 前記制御転送命令発行済み信号内に存在する前記微候に基づき別々のアドレス可能な記憶場所内に各々の分岐命令又はジャンプ&リンク命令について1つの情報項目を記憶するための手段；

を含んで成り、前記命令が分岐命令である場合、前記分岐命令のための前記情報項目が代替次プログラムカウンタを含み、前記命令がジャンプ&リンク命令である場合、ジャンプ&リンク命令のための前記情報項目が予測された標的アドレスを含んでいる、装置。

197. 発行された命令の誤予測が発生した後、命令再フェッチのための適正な再フェッチ命令アドレスを送るための方法において

— 前記命令の実行に先立ち各々の投機的に発行された予測された命令について、この予測された命令のための代替的実行経路を指定する代替的経路プログラム命令アドレス及び、前記CPU内のメモリユニット内に規定された標的アドレス構造内のウォッチポイント要素の中で前記予測された実行を左右する予測情報を、同時に記憶するための手段；

— 各々の前記予測された命令の実行の完了時点で、中に実行前代替経路及び予測情報が記憶されている前記命令に対応するウォッチポイント要素番号及び前記

記憶された条件値及び代替的プログラム命令アドレスとの比較のための計算上の適正な条件値を識別する、命令実行完了状況を表示する信号を同時に生成するための手段；

— 前記記憶された実行前代替経路及び予測情報と各単一命令のための前記生成された計算上の条件データ値とを比較することを含め各単一命令についての予測を同時に評価し、分岐命令又はジャンプ

&リンク命令が正しく予測されたか又は予測が正しくないかを評価し各々の誤予測命令について誤予測信号を生成するための手段；

— 前記誤予測からの前記CPUの回復を優先順位づけし、予め定められた優先順位規則に基づいて特定の高優先順の誤予測を選択するための手段；

— 前記特定の高優先順誤予測命令に付随するチェックポイントを識別し、前記特定の誤予測命令から前記プロセッサを回復するためのCPUバックアップ信号を生成するための手段；

— 前記標的アドレスデータ構造から前記特定の誤予測命令のための代替次プログラムカウンタを読取るための手段；及び

— 前記再フェッチされた命令に基づき前記CPUの実行を再開させるため発行ユニットに対して前記読取り代替次プログラムアドレスを送る段階、

を含んで成る装置：

198. — 前記同時に記憶するための手段には、前記ウォッチポイント要素への書込み及びここからの読取りのためのウォッチポイント読取り／書込み制御論理及び複数のウォッチポイント記憶要素が含まれており、これらのウォッチポイント要素は命令発行ユニットに結合され、各々の予測された命令のための命令復号情報を受理し；

— 前記命令実行完了信号を同時に生成するための手段には、データ順方向送りバスからの条件コードデータを含む実行結果データを捕捉するため複数の条件コード捕捉ユニットに結合された少なくとも1つの命令実行ユニットが含まれており；

— 前記同時に評価するための手段には、予測された命令を左右する予測された

及び実際の条件データを評価するための複数の評価論理ユニットが含まれており；

ー 前記優先順位づけ用手段には、実行状況を含む命令状況をトラッキングし予め定められた規則に基づき多重の誤予測及び実行障害から前記高優先順位の誤予測を選択するための精確な状態のユニットが含まれており；

ー 前記識別のための手段には、標的アドレスランダムアクセスメモリユニット、このメモリに付随する周辺制御論理；制御入力信号を有するマルチプレクサ回路及びバックアップチェックポイント入力そして分岐命令に付随するアドレスか又は前記標的アドレスランダムアクセスユニットからのジャンプ&リンク命令に付随するアドレスのいずれかをゲートするためのアドレス生成ユニットチェックポイント信号を含む複数の選択可能な信号入力端が含まれており；

ー 前記実行前条件を実際の実行条件結果と比較するためのジャンプ&リンカー致論理が含まれており；

ー 前記送信用手段には、標的アドレスランダムアクセスメモリ入力信号及び実行結果データ入力信号を有するマルチプレクサ回路及び複数のプログラムカウンタの中から適切な命令プログラムカウンタアドレスを出力するため前記精確な状態ユニットに結合された制御信号入力端を含む代替プログラムカウンタ出力論理が含まれている；

請求項197 に記載の装置。

199. 中央処理ユニット内での比較時間を短縮するようにデータ順方向送りバスタグと条件コードデータを比較するための装置において、

ー ・ 現行の発行ウインドウ信号内の現行のリネーム済み条件コードを各々のデータ順方向送りバスタグに比較するための手段；

ー ・ 先行するサイクルの条件コードタグを各々のデータ順方向送りバスタグと比較するための手段；

によって、先行するサイクル内の条件コードタグと現行の発行ウインドウサイクル内の現在リネームされている条件コードタグの間でデータ順方向送りタグの比

較を分離する手段を含み、

ー 先行するサイクル中の前記条件コードタグ及び現行の発行ウインドウ内の現在リネームされている条件コードタグとの前記データ順方向送りバスタグの前記比較は、単一の非分割作業で行なわれた場合に比べさらに迅速に行なわれるような装置。

200. 中央処理ユニット内で複数の命令の実行結果を同時に監視するための装置において、

- ー 命令を発行するための発行ユニット；
- ー 命令を実行するための実行ユニット；
- ー 前記中央処理ユニット内で前記実行ユニットからその他のユニットまで実行結果を伝達するデータ順方向分配バス；
- ー プロセッサ内で前記データ順方向分配の上で前記実行ユニットからの実行結果信号を受理するべく結合されたウォッチポイントデータを記憶するため複数のウォッチポイントレジスタを有するウォッチポイントユニット；
- ー 前記命令が発行された時点で各々の投機的発行済みの予測された命令についてウォッチポイントデータを記憶するためのウォッチポイントレジスタを割振るための手段；
- ー 前記ウォッチポイントユニット内で前記データ順方向送りバス上で伝達された実行結果信号を監視するための手段；
- ー 前記実行結果信号及び予め定められた規則に基づいて予め定められた事象の発生を検出するための手段；
- ー 予測された投機的発行済み命令について予め定められた事象が検出された時点でその他のCPU ユニットにそれを知らせるための手段；
- ー 特定の命令について前記ウォッチポイントレジスタ内に記憶された前記ウォッチポイントデータとこの特定の命令に関する前記データ順方向送りバスの上で到着した前記結果信号と比較して、この特定の命令が投機的に発行された条件が正しく予測されたか否かを評価するための手段；
- ー 前記予測が適正でなかったことを前記評価が表示した場合に、前記ウォッチ

ポイント要素を割振り解除し実行を続行するための割振り解除手段；

ー 前記予測が誤っていたということを前記評価が表示した場合に誤予測を補正するべく既知の条件に対応する命令を再フェッチし発行できるように、前記発行ユニットに対し新しいアドレスを供給するための手段；

ー 前記誤予測を取消すことができるように、前記中央処理ユニットをより早期の状態までバックアップするための手段；

ー 前記再フェッチされた命令に基づいて適正な命令ストリーム内の命令について実行を開始するための手段；及び

ー 前記実行が完了した時点で前記ウォッチポイント要素を割振り解除するための手段；

ー 各々の結果データソースから有効化制御信号を受理するための手段；

ー 各々の結果データソースから結果データタイプ信号を受理するための手段；

ー 前記実行結果生成ユニットのうちのいずれから前記結果データが受理されることになるかを表示するセレクトソース信号及び前記結果データが予想されている現行クロックサイクルとの関係における時刻の表示を受理するための手段；

ー 予め定められた規則に従って前記有効化制御信号、前記結果デ

ータタイプ信号及び前記セレクトソース信号に応じてソース選択信号を生成するための手段；

ー 前記実行結果生成ユニットから実行結果データ信号を監視するための手段；

ー 前記到着した実行結果データ及び前記ソース選択信号に応じて前記選択されたソースから実行結果データを捕捉するための手段；及び

ー 前記プロセッサ内のデータ記憶装置内での評価のため前記捕捉された実行結果データを記憶するための手段、

ー 分岐命令を監視するためにウォッチポイントが形成されつつあることを表示する第1の状態又はジャンプ&リンク命令のためのウォッチポイントが形成されつつあることを表示する第2の状態を有するウォッチポイントジャンプ&リンク有効化信号を受理するための手段；

ー ・条件コードセレクト論理回路からの評価条件コード信号を受理するための

手段；

- ・ウォッチポイント記憶要素からのウォッチポイント条件信号を受理するための手段；

- ・ウォッチポイント記憶要素からの条件コードタイプ制御信号を受理するための手段；

- ・各々の実行結果生成ユニットからの有効化信号を受理するための手段；

- ・分岐のための条件が正しく予測されたか否かを評価するため各々の前記実行結果生成ユニットについて前記有効化制御信号の表明された状態を含む予め定められた条件コード評価規則に従って前記ウォッチポイント条件信号と前記評価条件コード信号を比較するための手段；

- ・前記分岐が正しく予測されていた場合、分岐真信号を生成するための手段；を含む、複数の発行済み分岐命令について誤予測信号を同時に生成するための手段；及び

- ・プログラムカウンタデータを記憶するための複数の記憶場所を有する標的データ構造を規定するための手段；

- ・ジャンプ&リンク命令が発行された時点で前記記憶場所のうちの 1 つに代替次プログラムカウンタを書込み、前記記憶された代替次プログラムカウンタを前記命令と結びつけるための手段；

- ・実行結果生成ユニット内で計算上の次プログラムカウンタ値を計算するための手段；

- ・前記実行結果生成ユニットから前記計算上の次プログラムカウンタを受理するための手段；

- ・前記記憶場所からの前記命令に付随する前記記憶された代替次プログラムカウンタを読取るための手段；

- ・予め定められたプログラムカウンタ比較規則に従って前記計算上の次プログラムカウンタと前記代替次プログラムカウンタを比較するための手段；

- ・前記比較が真として評価した場合、ジャンプ&リンケー致信号を生成するための手段；及び

・前記ジャンプ&リンカー致信号と前記ウォッチポイントジャンプ&リンク信号の両方の真の状態に応じて、ジャンプ&リンク真信号を生成するための手段、を含む、複数の発行済みジャンプ&リンク命令の各々についてジャンプ&リンク誤予測信号を同時に生成するための手段；

ー 前記ジャンプ&リンカー致信号が表明されるか又は前記分岐真信号が表明されるかのいずれかに応えて、評価真信号を生成するた

めの手段；

ー 少なくとも1つのソース選択信号の受理又は評価有効化信号の受理に応じて評価準備完了信号を生成するための手段であって、この評価有効化信号はウォッチポイント活動中信号、ウォッチポイントジャンプ&リンク有効化信号、ジャンプ&リンク命令実行完了信号及びジャンプ&リンク命令チェックポイント番号信号の表明に応じて生成されるような手段；及び

ー 前記評価準備完了信号及び前記評価真信号の受理に応じて命令誤予測信号を生成するための手段

ー 各々の発行された制御転送命令についてそれが発行された時点で発行ユニットから制御転送命令発行済み信号を生成するための手段；

ー 分岐命令又はジャンプ&リンク命令として命令を識別するべく前記制御転送命令発行済み信号内でウォッチポイントジャンプ&リンク信号を提供するための手段；

ー プログラム制御論理ユニットにより計算された標的フェッチプログラムカウンタを特定する代替次プログラムカウンタを計算するための手段；

ー 前記データ構造内の記憶場所の中に代替分岐アドレスを記憶するためチェックポイント書込み有効化制御信号及びチェックポイントアドレスを生成するための手段であって、この書込み有効化制御信号がチェックポイントを作成するときに表明されるような手段；

ー チェックポイント書込みアドレスにおいて記憶ユニット内にウォッチポイント代替次プログラムカウンタを書込むための手段；

ー 予測された分岐及びジャンプ&リンク命令の命令復号の間に代替次プログラ

ムカウンタを決定するための手段であって、ジャンプ&リンク命令のためにウォッチポイントユニットに対して送られた

前記ウォッチポイント代替次プログラムカウンタには、予測された標的フェッチプログラムカウンタが含まれ、予測された分岐命令のためにウォッチポイントユニットに対し送られた前記ウォッチポイント代替次プログラムカウンタには、代替分岐方向のためのフェッチプログラムカウンタが含まれているような手段；

— 発行された命令が分岐命令であるかジャンプ&リンク命令であるかを決定することを含め、フィールド選択及び書込み論理ユニット内で前記制御転送命令発行済み信号を復号して前記信号内の選択された情報フィールドの内容を決定するための手段であって、前記JMPLフィールド内の第1の予め定められた状態は、命令がジャンプ&リンク命令であることを表示し、前記JMPLフィールド内の第2の状態は前記命令が分岐命令であることを表示しているような手段；及び

— 前記制御転送命令発行済み信号内に存在する前記徴候に基づき別々のアドレス可能な記憶場所内に各々の分岐命令又はジャンプ&リンク命令について1つの情報項目を記憶するための手段であって、前記命令が分岐命令である場合、前記分岐命令のための前記情報項目が代替次プログラムカウンタを含み、前記命令がジャンプ&リンク命令である場合、ジャンプ&リンク命令のための前記情報項目が予測された標的アドレスを含んでいるような手段；

— 前記命令の実行に先立ち各々の投機的に発行された予測された命令について、この予測された命令のための代替的実行経路を指定する代替的経路プログラム命令アドレス及び、前記CPU内のメモリユニット内に規定された標的アドレス構造内のウォッチポイント要素の中で前記予測された実行を左右する予測情報を、同時に記憶するための手段；

— 各々の前記予測された命令の実行の完了時点で、中に実行前代

替経路及び予測情報が記憶されている前記命令に対応するウォッチポイント要素番号及び前記記憶された条件値及び代替的プログラム命令アドレスとの比較のための計算上の適正な条件値を識別する、命令実行完了状況を表示する信号を同時

に生成するための手段；

－ 各々の前記命令についての前記生成された計算上の条件データ値と前記記憶された実行前代替経路及び予測情報を比較し、予測された条件データが前記命令のいずれかについて前記計算上の正しい条件データ値と一致する場合に一致信号を生成するための手段；

－ 分岐命令又はジャンプ&リンク命令が正しく予測されたか又は予測が正しくないかを評価し各々の誤予測命令について誤予測信号を生成することを含め、各々の単一命令についての予測を同時に評価するための手段；

－ 前記誤予測からの前記CPUの回復を優先順序づけし、予め定められた優先順位規則に基づいて特定の高優先順の誤予測を選択するための手段；

－ 前記特定の高優先順誤予測命令に付随するチェックポイントを識別し、前記特定の誤予測命令から前記プロセッサを回復するためのCPUバックアップ信号を生成するための手段；

－ 前記標的アドレスデータ構造から前記特定の誤予測命令のための代替次プログラムカウンタを読取るための手段；及び

－ 前記再フェッチされた命令に基づき前記CPUの実行を再開させるため発行ユニットに対して前記読取り代替次プログラムアドレスを送る段階、

－ ・現行の発行ウインドウ信号内の現行の再命令済み条件コードを各々のデータ順方向送りバスタグに比較するための手段；

・先行するサイクルの条件コードタグを各々のデータ順方向送りバスタグと比較するための手段；

によって、先行するサイクル内の条件コードタグと現行の発行ウインドウサイクル内の現在リネームされている条件コードタグの間でデータ順方向送りタグの比較を分離する手段であって

先行するサイクル中の前記条件コードタグ及び現行の発行ウインドウ内の現在リネームされている条件コードタグとの前記データ順方向送りバスタグの前記比較は、単一の非分割作業で行なわれた場合に比べさらに迅速に行なわれるような手段；

を含んで成る装置。

201. プログラム制御順out-of-order実行データプロセッサにおいて、

— 実行のためプログラム制御順で命令を発行するための発行ユニットであって、発行される命令には、浮動小数点及び非浮動小数点命令が含まれている、ユニット；

— 少なくとも浮動小数点命令が、実行手段によってプログラム制御順外でout-of-order実行され得る、発行済み命令を実行するための実行手段；

— ・発行済み命令の各々が記憶要素の1つに対応し、各々の記憶要素が浮動小数点命令識別フィールドと浮動小数点トラップタイプフィールドを有する、記憶要素を含むデータ記憶構造；

・各々の発行済み命令について対応する発行済み命令が浮動小数点命令であるか否かを表示する対応する記憶要素の浮動小数点命令識別フィールド内へのデータを書込むための第1の論理；

・実行中に浮動小数点実行トラップの予め規定された複数のタイプのうちの対応する1つのタイプを結果としてもたらすことになる単数又は複数の浮動小数点実行例外をひきおこす各々の発行された浮動小数点命令について、結果としてもたらされることになる浮動小数点実行トラップの予め定められたタイプのうちの1つを識別す

る対応する記憶要素の浮動小数点トラップタイプフィールド内へのデータを書込むための第2の論理

を含む浮動小数点例外ユニット；

— 実行中実行例外をひき起こさず、プログラム制御順でそれに先行する全ての発行済み命令が退去されてしまっている各々の発行済み命令を退去させるための精確状態手段；

を含み、

予め定められた実行例外のうちの第1のものが発行済み命令によってひき起こされた時点で、実行手段は発行済み命令の実行を続行し、精確状態手段は、退去され得ない発行済み命令に遭遇するまで発行済み命令を退去させ続けることによ

り実行トラップ順序付けに着手し、退去され得ない発行済み命令は、(a) 第1の実行例外をひき起こした発行済み命令、及び(b) 第1の実行例外をひき起こした発行済み命令よりも早期に発行されたものの第1の実行例外よりも晩期に発生する第2の実行例外をひき起こした発行済み命令のうちの1つであり、さらに

- 浮動小数点トラップタイプフィールドをもつ浮動小数点状況レジスタ；及び
- 退去され得ない命令に対応する記憶要素の浮動小数点識別フィールド内のデータが、この退去され得ない命令が浮動小数点命令であることを表示した時点で、退去され得ない命令に対応する記憶要素の浮動小数点トラップタイプフィールド内のデータにより識別された浮動小数点実行トラップのタイプを識別する浮動小数点状況レジスタの浮動小数点トラップタイプフィールドに対するデータを書込むための書込み手段；

を含んで成るプロセッサ。

202. — データ記憶構造の各々の記憶要素が、現行の浮動小数

点実行例外フィールドも有しており；

- 浮動小数点例外ユニットにはさらに、浮動小数点実行トラップの予め定められたタイプのうちの特定の1つのタイプを結果としてもたらすことになる複数の特定の浮動小数点実行例外のうちの単数又は複数の例外を実行中にひき起こす各々の発行済み浮動小数点命令について、ひき起こされた特定の浮動小数点実行例外のうちの単数又は複数のものを識別する対応する記憶要素の現行の浮動小数点実行例外フィールドに対するデータを書込むための第3の論理がさらに含まれ；
- 浮動小数点状況レジスタが同様に、現行の浮動小数点実行例外フィールドも有しており、さらに
- 書込み手段は同様に、(a) 退去され得ない発行済み命令に対応する記憶要素の浮動小数点命令識別フィールド内のデータが、退去され得ない発行済み命令が浮動小数点命令であることを表示した時点、及び(b) 退去され得ない発行済み命令に対応する記憶要素の浮動小数点トラップタイプフィールド内のデータが、予め定められた浮動小数点実行トラップタイプのうちの特定の1つを識別した時点で、退去され得ない発行済み命令に対応する記憶要素の現行の浮動小数点実

行例外フィールド内のデータによって識別された特定の浮動小数点実行例外のうちの単数又は複数のものを識別する浮動小数点状況レジスタの現行の浮動点実行例外フィールドに対するデータを書込む、

請求項201 に記載のプログラム制御順out-of-order実行データプロセッサ。

203. — 第4の論理は同様に、実行中に特定の浮動小数点実行例外のいずれもひき起こさない各々の発行済み浮動小数点命令について、特定の浮動小数点実行例外のいずれもひき起こされていない

ことを表示する対応する記憶要素の現行の浮動小数点実行例外フィールドに対するデータを書込み；

— 浮動小数点状況レジスタには、蓄積した例外フィールド及びトラップ有効化マスクフィールドが含まれ、トラップ有効化マスクフィールドが特定の浮動小数点実行例外のあらゆる組合せの選択的マスキングを提供し；

— 精確状態ユニットは同様に、実行中単数又は複数のマスキングされた特定の浮動小数点実行例外をひき起こすもののその他の浮動小数点実行例外を全くひき起こさず、かつプログラム制御順でそれに先行する全ての発行済み命令が退去させられてしまっている各々の発行済み浮動小数点命令も退去させ；

— 命令が退去させられる各々の現行のマシンサイクルの間、書込み手段は、（a）対応する記憶要素の浮動小数点命令識別フィールド内のデータが浮動小数点命令でありかつ現行のマシンサイクル内で退去されつつある発行済み命令に対応する記憶要素の現行の浮動小数点実行例外フィールド内のデータによって識別された特定の浮動小数点実行例外、及び（b）浮動小数点状況レジスタの蓄積した浮動小数点実行例外フィールド内の現行データによって識別された特定の浮動小数点実行例外の蓄積を表わす浮動小数点状況レジスタの蓄積された例外フィールドに対するデータを書込み；

— 命令が退去させられる各々の現行のマシンサイクルの間に、書込み手段は同様に対応する記憶要素の浮動小数点命令識別フィールド内のデータがそれが浮動小数点命令であることを表示している対応する現行のマシンサイクルにおいて退去させられた最後に発行された命令に対応する記憶要素の現行の浮動小数点実行

例外フィールド内のデータによって識別された特定の浮動小数点実行例外を識別する浮動小数点状況レジスタの現行の例外フィールドに対するデータ

タを書込み；

— 命令が退去させられる各々の現行のマシンサイクルの間に、書込み手段はさらに、対応する記憶要素の浮動小数点命令識別フィールド内のデータがそれが浮動小数点命令であることを表わしている対応する現行のマシンサイクルにおいて退去させられた最後に発行された命令について、予め定められた浮動小数点実行トラップタイプのいずれも結果としてもたらされないであろうということを表示する浮動小数点状況レジスタのトラップタイプフィールドに対するデータを書込む

請求項202 に記載のプログラム制御順out-of-order実行データプロセッサ。

204. 発行された命令にはSPARC 命令が含まれており；

— 浮動小数点状況レジスタが、SPARC 浮動小数点状況レジスタを含んでおり、
— 浮動小数点実行トラップの予め定められたタイプのうちの特定の1つのタイプがIEEE_754_ 例外トラップを含み、
— 特定の浮動小数点実行例外には、IEEE_754例外が含まれる、

請求項203 に記載のプログラム順out-of-order実行のデータプロセッサ。

205. プログラム制御順out-of-order実行データプロセッサ内で浮動小数点例外を検出する方法において、

— 実行のためプログラム制御順で命令を発行する段階であって発行される命令には、浮動小数点及び非浮動小数点命令が含まれている段階；
— 少なくとも浮動小数点命令が、実行手段によってプログラム制御順外でout-of-order実行され得るように発行済み命令を実行する段階；

— ・発行済み命令の各々が記憶要素の1つに対応し、各々の記憶要素が浮動小数点命令識別フィールドと浮動小数点トラップタイプフィールドを有する、記憶要素を含むデータ記憶構造を提供する段階；

・各々の発行済み命令について対応する発行済み命令が浮動小数点命令である

か否かを表示する対応する記憶要素の浮動小数点命令識別フィールド内へのデータを書込む段階；

・ 実行中に浮動小数点実行トラップの予め規定された複数のタイプのうちの対応する1つのタイプを結果としてもたらすことになる単数又は複数の浮動小数点実行例外をひき起こす各々の発行された浮動小数点命令について、結果としてもたらされることになる浮動小数点実行トラップの予め定められたタイプのうちの1つを識別する対応する記憶要素の浮動小数点トラップタイプフィールド内へのデータを書込む段階；

－ 実行中実行例外をひき起こさず、プログラム制御順でそれに先行する全ての発行済み命令が退去されてしまっている各々の発行済み命令を退去させる段階；

－ 予め定められた実行例外のうちの第1のものが発行済み命令によってひき起こされた時点で、発行済み命令の実行を続行し、退去され得ない発行済み命令に遭遇するまで発行済み命令を退去させ続けることにより実行トラップ順序付けに着手し、ここで退去され得ない発行済み命令は、(a) 第1の実行例外をひき起こした発行済み命令、及び(b) 第1の実行例外をひき起こした発行済み命令よりも早期に発行されたものの第1の実行例外よりも晩期に発生する第2の実行例外をひき起こした発行済み命令のうちの1つである段階；

－ 浮動小数点トラップタイプフィールドをもつ浮動小数点状況レ

ジスタを提供する段階；及び

－ 退去され得ない命令に対応する記憶要素の浮動小数点識別フィールド内のデータか、この退去され得ない命令が浮動小数点命令であることを表示した時点で、退去され得ない命令に対応する記憶要素の浮動小数点トラップタイプフィールド内のデータにより識別された浮動小数点実行トラップのタイプを識別する浮動小数点状況レジスタの浮動小数点トラップタイプフィールドに対するデータを書込む段階；

を含んで成る方法。

206. － データ記憶構造の各々の記憶要素が、現行の浮動小数点実行例外フィールドも有しており；

ー 浮動小数点状況レジスタが同様に、現行の浮動小数点実行例外フィールドも有しており、さらに

ー 浮動小数点実行トラップの予め定められたタイプのうちの特定の1つのタイプを結果としてもたらしことになる複数の特定の浮動小数点実行例外のうちの単数又は複数の例外を実行中に引き起こす各々の発行済み浮動小数点命令について、引き起こされた特定の浮動小数点実行例外のうちの単数又は複数のものを識別する対応する記憶要素の現行の浮動小数点実行例外フィールドに対するデータを書込む段階；

ー (a) 退去され得ない発行済み命令に対応する記憶要素の浮動小数点命令識別フィールド内のデータが、退去され得ない発行済み命令が浮動小数点命令であることを表示した時点、及び(b) 退去され得ない発行済み命令に対応する記憶要素の浮動小数点トラップタイプフィールド内のデータが、予め定められた浮動小数点実行トラップタイプのうちの特定の1つを識別した時点で、退去され得ない発行済み命令に対応する記憶要素の現行の浮動小数点実行例外フ

ィールド内のデータによって識別された特定の浮動小数点実行例外のうちの単数又は複数のものを識別する浮動小数点状況レジスタの現行の浮動点実行例外フィールドに対するデータを書込む段階；

を含んで成る請求項205 に記載の方法。

207. ー 実行中に特定の浮動小数点実行例外のいずれも引き起こさない各々の発行済み浮動小数点命令について、特定の浮動小数点実行例外のいずれも引き起こされていないことを表示する対応する記憶要素の現行の浮動小数点実行例外フィールドに対するデータを書込む段階を含み；

ー ここで浮動小数点状況レジスタには、蓄積した例外フィールド及びトラップ有効化マスクフィールドが含まれ、トラップ有効化マスクフィールドが特定の浮動小数点実行例外のあらゆる組合せの選択的マスキングを提供し；さらに

ー 実行中単数又は複数のマスキングされた特定の浮動小数点実行例外を引き起こすもののその他の浮動小数点実行例外を全く引き起こさず、かつプログラム制御順でそれに先行する全ての発行済み命令が退去させられてしまっている各々の

発行済み浮動小数点命令を退去させる段階；

ー 命令が退去させられる各々の現行のマシンサイクルの間、(a) 対応する記憶要素の浮動小数点命令識別フィールド内のデータが浮動小数点命令であり、かつ現行のマシンサイクル内で退去されつつある発行済み命令に対応する記憶要素の現行の浮動小数点実行例外フィールド内のデータによって識別された特定の浮動小数点実行例外、及び(b) 浮動小数点状況レジスタの蓄積した浮動小数点実行例外フィールド内の現行データによって識別された特定の浮動小数点実行例外の蓄積を表わす浮動小数点状況レジスタの蓄積された例外フィールドに対するデータを書込む段階；

ー 命令が退去させられる各々の現行のマシンサイクルの間に、対応する記憶要素の浮動小数点命令識別フィールド内のデータがそれが浮動小数点命令であることを表示している対応する現行のマシンサイクルにおいて退去させられた最後に発行された命令に対応する記憶要素の現行の浮動小数点実行例外フィールド内のデータによって識別された特定の浮動小数点実行例外を識別する浮動小数点状況レジスタの現行の例外フィールドに対するデータを書込む段階；

ー 命令が退去させられる各々の現行のマシンサイクルの間に、対応する記憶要素の浮動小数点命令識別フィールド内のデータがそれが浮動小数点命令であることを表わしている対応する現行のマシンサイクルにおいて退去させられた最後に発行された命令について、予め定められた浮動小数点実行トラップタイプのいずれも結果としてもたらされないであろうということを表示する浮動小数点状況レジスタのトラップタイプフィールドに対するデータを書込む段階；

をさらに含んで成る請求項206 に記載の方法。

208. 発行された命令にはSPARC 命令が含まれており；

ー 浮動小数点状況レジスタが、SPARC 浮動小数点状況レジスタを含んでおり、
ー 浮動小数点実行トラップの予め定められたタイプのうち特定の1つのタイプがIEEE_754_ 例外トラップを含み、

ー 特定の浮動小数点実行例外には、IEEE_754例外が含まれる、

請求項207 に記載の方法。

209. 投機的にトラップを取りこれから復帰するためのデータプロセッサにおいて、各々対応するトラップレベルを有するネストされたトラップを取るため予め定められた数のトラップレベルを支持するデータプロセッサであって、

- ー チェックポイントを形成するための手段；
- ー チェックポイントにバックアップするための手段；
- ー トラップを取るための手段；
- ー トラップから復帰するための手段；
- ー トラップが取られる度毎のデータプロセッサの状態を規定する内容をもつレジスタ
- ー ・トラップレベルより多い数のトラップスタック記憶エントリを有するトラップスタックデータ記憶構造；

・トラップレベルのうちの1つにマッピングするため現在利用可能なトラップスタック記憶エントリの現行の利用可能性リストを維持し、トラップがとられる毎にトラップレベルのうちの対応するものにマッピングするため現在利用可能なトラップスタック記憶エントリのうちの次のものを識別するフリーリストユニット；

・とられた各々のトラップについて、現在利用可能なトラップスタック記憶エントリのうちの次のものに対しレジスタの内容を書込む読取り／書込み論理；

・トラップスタック記憶エントリのうちの1つに対する各トラップレベルの現行のマッピングを維持し、トラップがとられる毎に、トラップスタック記憶エントリのうちの1つに対する対応するトラップレベルの古いマッピングを、現在利用可能なトラップスタック記憶エントリのうちの次のエントリに対する対応するトラップレベルの現行マッピングと置換する、リネームマッピング論理；

・現行マッピングによりトラップレベルの1つに現在マッピングされていないもののトラップレベルの1つにマッピングするためには利用できない各々のトラップスタック記憶エントリを利用不能性リストを維持し、トラップがとられる毎に、古いマッピングにより対応するトラップレベルにマッピングされたトラップスタック記憶エントリを利用不能性リストに付加し、とったトラップをもはや取

消しできなくなる毎に、古いマッピングにより対応するトラップレベルにマッピングされたトラップスタック記憶エントリを利用不能性リストから除去する資源再生ユニット；及び

- ・チェックポイント記憶エントリを含み、形成された各チェックポイントが対応するチェックポイント記憶エントリをもち、さらに各々の形成されたチェックポイントについて対応するチェックポイント記憶エントリ内でフリーリストユニットの現行の利用可能性フリーリスト及びリネームマッピング論理の現行のマッピングを記憶する、チェックポイント記憶ユニット、を含むトラップスタックユニット、

を含んで成り、

- ・フリーリストは、現行の利用可能性リストに対し、利用不能性リストから除去された各々のトラップスタック記憶エントリを付加し、

- ・チェックポイントに対する各々のバックアップについて、リネームマッピング論理は、それが維持する現行マッピングを対応するチェックポイント記憶エントリ内に記憶されているマッピングと置換し、フリーリストユニットはそれが維持する現行の利用可能性リストを対応するチェックポイント記憶エントリ内に記憶された利用可能性リストと置換する、

データプロセッサ。

210. — トラップがとられる毎に、レジスタのうちの特定のものの内容がまだ利用可能でない可能性があり；

— データプロセッサにはさらに、レジスタのうちの特定のものの内容が利用可能となった時点をとらわれる毎に決定するための捕捉論理が含まれており；

— レジスタのうちの特定の1つのレジスタの内容が利用可能にな

ったことを捕捉論理が決定した時点で、トラップがとられる毎に、この特定のレジスタの内容を利用可能なトラップスタック記憶エントリのうちの次のものに書込む、

請求項209に記載のトラップスタックユニット。

211. — 捕捉論理はレジスタのうちの特定のものの内容がまだ利用可能でないかどうかをトラップからの復帰のための手段に対し表示し、

— トラップから復帰するための手段は、レジスタのうちの特定のレジスタの内容が利用可能な状態になったことを捕捉論理が表示するまで、トラップから復帰しない、

請求項210 に記載のデータプロセッサ。

212. — フリーリストユニットは、トラップをとるための手段に対して、トラップスタック記憶エントリのうちの少なくとも1つが、トラップレベルへのマッピングのために現在利用可能であるか否かを表示し、

— トラップをとるための手段は、トラップスタック記憶エントリのうちの少なくとも1つがトラップレベルへのマッピングのために利用可能であることをフリーリストが表示するまで、トラップをとらない、

請求項209 に記載のデータプロセッサ。

213. — 形成されたチェックポイントを退去する段階；
をさらに含み、

— チェックポイント形成手段が、とられた各々のトラップについての対応するチェックポイントを形成し；

— 資源再生ユニットには、資源再生記憶エントリを有する資源再生データ記憶構造が含まれ、各々の形成されたチェックポイントは資源再生記憶エントリのうちの1つに対応し、資源再生ユニットは

、とられた各々のトラップについて、対応するトラップのために形成されたチェックポイントに対応する資源再生記憶エントリ内にデータを記憶し、このデータは、古いマッピングにより対応するトラップレベルに対しマッピングされたトラップスタック記憶エントリを識別し、資源再生ユニットは、チェックポイントを退去させるための手段によって、とられたトラップのために形成されたチェックポイントが退去される毎に、退去されたチェックポイントに対応する資源再生記憶エントリ内に記憶されたデータによって識別されたトラップスタック記憶エントリを識別し、

ー フリーリストユニットは、チェックポイントを退去させるための手段によって、とられたトラップのために形成されたチェックポイントが退去される毎に、資源再生ユニットによって識別された記憶エントリを、それが維持する現行の利用可能性リストに対して付加する、

請求項209 に記載のデータプロセッサ。

214. 対応するトラップレベルを各々有するネストされたトラップをとるため予め定められた数のトラップレベルを支持するデータプロセッサの中で、投機的にトラップをとりこれから復帰する方法であって、

- ー チェックポイントを形成する段階；
- ー チェックポイントをバックアップする段階；
- ー トラップを取る段階；
- ー トラップから復帰する段階；
- ー トラップが取られる度毎のデータプロセッサの状態を規定する内容をもつレジスタを提供する段階；
- ー トラップレベルより多い数のトラップスタック記憶エントリを有するトラップスタックデータ記憶構造を提供する段階；
- ー トラップレベルのうちの1つにマッピングするため現在利用可能なトラップスタック記憶エントリの現行の利用可能性リストを維持し、トラップがとられる毎にトラップレベルのうちの対応するものにマッピングするため現在利用可能なトラップスタック記憶エントリのうちの次のものを識別する段階；
- ー とられた各々のトラップについて、現在利用可能なトラップスタック記憶エントリのうちの次のものに対しレジスタの内容を書込む段階；
- ー トラップがとられる毎に、トラップスタック記憶エントリのうちの1つに対する対応するトラップレベルの古いマッピングを、現在利用可能なトラップスタック記憶エントリのうちの次のエントリに対する対応するトラップレベルの現行マッピングと置換することによってトラップスタック記憶エントリのうちの1つに対する各トラップレベルの現行のマッピングを維持する段階；
- ー トラップがとられる毎に、古いマッピングにより対応するトラップレベルに

マッピングされたトラップスタック記憶エントリを利用不能性リストに付加し、
とったトラップをもはや取消しできなくなる毎に、古いマッピングにより対応する
トラップレベルにマッピングされたトラップスタック記憶エントリを利用不能
性リストから除去することによって、現行マッピングによりトラップレベルの1
つに現在マッピングされていないもののトラップレベルの1つにマッピングする
ためには利用できない各々のトラップスタック記憶エントリを利用不能性リスト
を維持する段階；

- ー 現行の利用可能性リストに対し、利用不能性リストから除去された各々のトラップスタック記憶エントリを付加する段階；
- ー チェックポイント記憶エントリを含み、形成された各チェックポイントが対応するチェックポイント記憶エントリをもつような、

チェックポイント記憶ユニットを提供する段階；

- ー 各々の形成されたチェックポイントについて、対応するチェックポイント記憶エントリ内で、現行のマッピング及び現行に利用可能性リストを記憶する段階；
- ー チェックポイントに対する各々のバックアップについて、現行のマッピングを対応するチェックポイント記憶エントリ内に記憶されたマッピングと置換し、現行の利用可能性リストを対応するチェックポイント記憶エントリ内に記憶された利用可能性リストと置換する段階、
を含んで成る方法。

215. ー トラップがとられる毎に、レジスタのうちの特定のものの内容がまだ利用可能でない可能性があり；

- ー レジスタのうちの特定のものの内容が利用可能となった時点で、トラップが取られる毎にこれを決定する段階；
- ー レジスタのうちの特定の1つのレジスタの内容が利用可能になったことが決定された時点で、トラップがとられる毎に、この特定のレジスタの内容を利用可能なトラップスタック記憶エントリのうちの次のものに書込む段階、
を含んで成る請求項214に記載の方法。

216. — レジスタのうちの特定のものの内容がまだ利用可能でないかどうかを表示する段階；

— 表示段階中でレジスタのうちの特定のものの内容が利用可能になったことが表示された時点で初めてトラップから復帰する段階；

をさらに含んで成る請求項215 に記載の方法。

217. — トラップスタック記憶エントリのうちの少なくとも1つがトラップレベルへのマッピングのために現在利用可能であるか否かを表示する段階；

— トラップスタック記憶エントリのうちの少なくとも1つがトラップレベルへのマッピングのために利用可能であることが、表示段階で表示された時点で初めてトラップをとる段階；

をさらに含んで成る、請求項214 に記載の方法。

218. — 形成されたチェックポイントを退去する段階；

をさらに含み、

— チェックポイント形成段階が、とられた各々のトラップについての対応するチェックポイントを形成する段階を含み；

— 利用不能性リストを維持する段階には、

・資源再生記憶エントリを有する資源再生データ記憶構造を提供し、ここで各々の形成されたチェックポイントが資源再生記憶エントリのうちの1つに対応し、資源再生ユニットは、記憶を行なうような段階；

・とられたトラップの各々について、対応するトラップのために形成されたチェックポイントに対応する資源再生記憶エントリ内にデータを記憶し、このデータは、古いマッピングにより対応するトラップレベルに対しマッピングされたトラップスタック記憶エントリを識別するような段階、

・チェックポイントを退去させるための手段によって、とられたトラップのために形成されたチェックポイントが退去される毎に、退去されたチェックポイントに対応する資源再生記憶エントリ内に記憶されたデータによって識別されたトラップスタック記憶エントリを識別する段階；

が含まれており、

ー 付加段階には、チェックポイントを退去させるための手段によって、とられたトラップのために形成されたチェックポイントが退去される毎に、資源再生ユニットフリーエントリによって識別され

た記憶エントリを、それが維持する現行の利用可能性リストに対して付加する段階が含まれている、

請求項214 に記載の方法。

219. 命令を発行するための手段、命令を実行するための手段及びプロセッサの中にデータを記憶するためのメモリ記憶装置を有するプロセッサの中で、プロセッサ状態回復を要求する条件を検出した時点であらゆる命令境界でプロセッサ内の先行するマシン状態を回復する方法において、

- ー 各々の発行済み命令について1つの命令シリアル番号を割振る段階；
- ー マシン状態を変更する実行可能な命令の第1の予め定められたセットについてのみ実行に先立って前記プロセッサ内のデータ記憶装置内で1つのチェックポイント中にマシン状態情報を記憶する段階であって、ここでマシン状態を変える実行可能な命令の前記予め定められたセットが、このプロセッサ内で実施されたマシン状態を変化させる実行可能な命令の全てのセットよりも少ないような段階；
- ー 前記障害をひき起こす命令を識別し、多数の例外又は障害が同時に発生した場合には、障害又は例外をひき起こす順序的に最も早期のin-order命令を識別する段階；
- ー 前記最も早期の障害発生命令がチェックポイント実行された命令である場合には、前記障害発生命令の実行に先立って記憶された前記マシン状態情報を回復させプロセッサプログラムカウンタを前記障害発生命令の命令シリアル番号まで減分させて前記先行するマシン状態を回復する段階；
- ー マシン状態情報が記憶されてきた前記予め定められた実行可能な命令の1つが順序的に前記障害発生命令と最後に発行された命令

の間に置かれている場合には、(i) まず最初に、順序的に前記障害発生命令の

後で前記障害発生命令に最も近いチェックポイントまで前記プロセッサをバックアップし、(ii) 第2に、レジスタ資源を更新することによって前記障害命令の実行の直前に存在した状態までプロセッサの状態を回復するように前記プロセッサをバックステップさせ、(iii) 前記障害発生命令の命令シリアル番号までプロセッサプログラムカウンタを減分させる段階；
を含んで成る方法。

220. 前記プロセッサ内のデータ記憶装置内にマシン状態情報を記憶する前記段階には、レジスタデータ値を記憶するよりもむしろレジスタリネームマップを記憶し、投機的に発行された予測された命令について代替次プログラムカウンタを記憶する段階、が含まれている請求項219に記載の方法。

221. 前記実行可能な命令の第1の予め定められたセットが、
— 予測された分岐命令を含む予測されたプログラム制御転送命令；
— 制御レジスタ値を修正する副作用をもち得る命令；
— ジャンプ&リンクタイプの命令を含め、プログラムフロー変化を結果としてもたらすタイプの命令；
— チェックポイント実行済み状態の量を減少させるためそのタイプの全ての命令よりも少ない数に制限され、そのタイプの非チェックポイント実行済み命令が結果として制御されたプロセッサ同期化をもたらすことになるような、制御レジスタ値を変更する副作用をもち得る命令を含めた、プロセッサ状態を修正するタイプの選択された命令；
— 予め定められた基準に従って頻繁に発生し、命令発行トラップを結果としてもたらす命令；及び

— 投機的トラップエントリを開始させるタイプの命令
から成るグループの中から選択された命令である請求項219に記載の方法。

222. 投機的命令シーケンスが関与する実行可能な命令の第2の予め定められたセットのうちのいずれか1つを実行する前に前記プロセッサを同期化する段階をさらに含む、請求項220に記載の方法。

223. 前記マシンの同期化段階には、

ー 実行に先立ちマシンの同期化を要求する前記その他の命令の発行／実行を暫定的に中断する段階；

ー 完遂し退去させるべき実行段において全ての保留命令を待ち、前記プロセッサがマシン同期化に到着した後で前記その他の命令を発行／実行する段階、が含まれている請求項222 に記載の方法。

224. 命令を発行するための手段、命令を実行するための手段及びプロセッサ内でデータを記憶するためのメモリ記憶装置を有するプロセッサの中で、命令境界においてプロセッサ内でマシン状態を回復する方法において、

ー 予め定められた複数の投機的発行済み予測命令の実行に先立って前記プロセッサの前記データ記憶装置内のレジスタに、予め定められたマシン状態パラメータを記憶する段階；

ー 実行例外又は障害条件を検出するべく命令実行を監視する段階；

ー 前記実行例外又は障害条件を結果としてもたらす各命令の命令識別子をセーブする段階；

ー 複数の実行済み命令が実行例外又は障害条件を結果としてもたらした場合に、この複数の例外又は障害発生命令のいずれが順序的

により早期であるかを決定する段階；

ー 前記最も早期の例外又は障害発生識別子をエンドポイントとして用いてプロセッサ中のプログラムカウンタをバックアップする段階；

ー 前記最も早期の障害発生命令がチェックポイント実行済み命令である場合に、この障害発生命令の実行に先立ち前記チェックポイント内に記憶された前記マシン状態情報を回復し、プロセッサプログラムカウンタを前記障害発生命令の命令シリアル番号まで減分して前記先行するマシン状態を回復する段階；及び

ー マシン状態情報が記憶された前記予め定められた実行可能な命令の1つが、順序的に前記障害発生命令と最後に発行された命令の間に置かれている場合には、(i) まず最初に、前記障害発生命令の後で前記障害発生命令に最も近いチェックポイントまで前記プロセッサをバックアップし、(ii) 第2にレジスタ資源を更新することによって前記障害命令の実行の直前に存在した状態までプロセッ

サの状態を回復するように前記プロセッサをバックアップさせ、(iii) 前記障害発生命令の命令シリアル番号までプロセッサプログラムカウンタを減分させる段階；

を含んで成る方法。

225. 命令境界でマシン状態を回復する方法において；

- ー 制御レジスタ値を修正する副作用をもち得る命令を含め、アーキテクチャ制御レジスタを修正し得るあらゆる命令について、(i) 前記マシンを同期化すること又は(ii) 制御レジスタ更新が必要でなくなるように前記命令の実行に先立ってマシン状態を保存するべく前記プロセッサ内の前記データ記憶構造内で命令チェックポイントを生成、記憶することのいずれかを選ぶ段階；
- ー 任意の投機的命令シーケンスを含むプログラムカウンタ不連続

性を作り出すあらゆる命令について、プログラムカウンタ値がin-order命令シーケンス内の命令に対応しかつ障害をひき起こした命令の再発行及び実行なく正しいプログラムカウンタを再構築できるように前記命令の実行に先立ってアーキテクチャ及びマシン状態を保持するべく前記プロセッサ内のデータ記憶装置内で命令チェックポイントを生成、記憶する段階；

- ー 実行例外又は障害条件を検出するべく命令実行を監視する段階；
- ー 前記実行例外又は障害条件を結果としてもたらす各命令の命令識別子をセーブする段階；
- ー 複数の実行済み命令が実行例外又は障害条件を結果としてもたらした場合に、この複数の例外又は障害発生命令のいずれが順序的により早期であるかを決定する段階；
- ー 前記最も早期の例外又は障害発生識別子をエンドポイントとして用いてプロセッサ中のプログラムカウンタをバックアップする段階；
- ー 前記最も早期の障害発生命令がチェックポイント実行済み命令である場合に、この障害発生命令の実行に先立ちチェックポイント内に記憶された前記マシン状態情報を回復し、プロセッサプログラムカウンタを前記障害発生命令の命令シリアル番号まで減分して前記先行するマシン状態を回復する段階；及び

ー マシン状態情報が記憶された前記予め定められた実行可能な命令の1つが、順序的に前記障害発生命令と最後に発行された命令の間に置かれている場合には、(i) まず最初に、前記障害発生命令の後で前記障害発生命令に最も近いチェックポイントまで前記プロセッサをバックアップし、(ii) 第2にレジスタ資源を更新することによって前記障害命令の実行の直前に存在した状態までプロセッサの状態を回復するように前記プロセッサをバックアップさせ、(iii) 前記障害発生命令の命令シリアル番号までプロセッサプログラムカウンタを減分させる段階；

を含んで成る方法。

226. 前記命令識別子は、前記命令が発行された時点で割当てられた前記命令シリアル番号である請求項225 に記載の方法。

227. 各々の前記チェックポイントには、レジスタリネームマップ及び制御；アーキテクチャ制御レジスタ値；ISN；アーキテクチャプログラムカウンタ（PC）及び次のアーキテクチャプログラムカウンタ（PC）；及び代替次プログラムカウンタ（PC）が含まれている、請求項226 に記載の方法。

228. 前記チェックポイントは、命令発行サイクル中に生成され割当てられる請求項225 に記載の方法。

229. 前記マシンの同期化段階には、

ー 実行に先立ちマシンの同期化を要求する前記その他の命令の発行／実行を暫定的に中断する段階；

ー 完遂し退去させるべき実行段において全ての保留命令を待ち、前記プロセッサがマシン同期化に到着した後で前記その他の命令を発行／実行する段階、

が含まれている請求項225 に記載の方法。

230. 実行に先立ちマシンの同期化を要求するために識別された前記その他の命令が、命令の同期化を要求するための性能劣化及び投機度への依存度といった考慮事項を含む性能－設計トレードオフに基づいて選択される、請求項225 に記載の方法。

231. 実行前の前記マシンの同期化を要求するものとしてマシン内のどの命令

を指定すべきかを選択する方法において、

- ー 前記命令のための投機度を決定する段階；
- ー 命令の全てを投機的にとり扱うのに必要となる要求された論理を実施するよう割振ることが望ましい最大回路部域を決定することを含む、実施することが望ましい論理的複雑性の限界を決定する段階；
- ー 各々のトレードオフパラメータについての重み係数を含め、予め定められたトレードオフパラメータを設定する段階；
- ー 前記決定及び前記基準に基づいてその命令が同期化を要求するはずのものであるか否かを決定するべく命令のための性能指標を計算する段階、
を含んで成る方法。

232. 命令を発行するための手段、命令を実行するための手段及びプロセッサの中にデータを記憶するためのメモリ記憶装置を有するプロセッサの中で、あらゆる命令境界でプロセッサ内の先行するマシン状態を回復するための装置において

- ー 各々の発行済み命令について1つの命令シリアル番号を割振るための手段；
- ー マシン状態を変更する実行可能な命令の第1の予め定められたセットについてのみ実行に先立って前記プロセッサ内のデータ記憶装置内で1つのチェックポイント中にマシン状態情報を記憶する手段であって、ここでマシン状態を変える実行可能な命令の前記予め定められたセットが、このプロセッサ内で実施されたマシン状態を変化させる実行可能な命令の全てのセットよりも少ないような手段；
- ー 前記障害をひき起こす命令を識別し、多数の例外又は障害が同時に発生した場合には、障害又は例外をひき起こす順序的に最も早期のin-order命令を識別する手段；
- ー 前記最も早期の障害発生命令がチェックポイント実行された命

令である場合には、前記障害発生命令の実行に先立って記憶された前記マシン状態情報を回復させプロセッサプログラムカウンタを前記障害発生命令の命令シリ

アル番号まで減分させて前記先行するマシン状態を回復するための手段；

－ マシン状態情報が記憶された前記予め定められた実行可能な命令の1つが順序的に前記障害発生命令と最後に発行された命令の間に置かれている場合に、より早期の状態まで前記プロセッサをバックトラッキングするための手段；を含み、

－ 前記バックトラッキング用手段には、順序的に前記障害発生命令の後に前記故障発生命令に最も近いチェックポイントまで前記プロセッサをバックアップするための手段が含まれ、しかも

－ レジスタ資源を更新することにより前記故障発生命令の実行の直前に存在していたプロセッサの状態を回復するべく、前記プロセッサをバックステップさせるための手段；及び

－ 前記障害発生命令の命令シリアル番号までプロセッサプログラムカウンタを減分させるための手段、
を含んで成る装置。

233. 前記プロセッサ内の前記データ記憶装置内にマシン状態情報を記憶するための前記手段には、前記マシン状態のためのレジスタリネームマップを記憶するための手段が含まれている請求項232 に記載の装置。

234. 前記レジスタリネームマップから前記状態を回復することによって前記マシン状態を回復するための手段をさらに含んで成る請求項233 に記載の装置。

235. 投機的命令シーケンスが関与する実行可能な命令の第2の予め定められたセットのいずれか1つを実行する前に前記プロセッサを同期化するための手段をさらに含んで成る請求項232 に記載の

方法。

236. 前記マシンの同期化用手段には、

－ 実行に先立ちマシンの同期化を要求する前記その他の命令の発行／実行を暫定的に中断するための手段；

－ 以前に発行された命令が完遂され退去されてしまうまで実行段階中の全ての保留命令の発行を遅延させるための手段；

ー 前記プロセッサがマシンの同期化に到達した後に前記同期化タイプの命令の発行及び実行を開始するための手段；

が含まれている請求項235に記載の方法。

237. データを記憶するための内部データ記憶装置、命令を発行するための命令発行ユニット手段、命令復号ユニット、命令を実行するための命令実行ユニット手段及び命令発行及び実行スケジューラ、及び前記実行ユニットから処理ユニット内のその他のユニットまで実行結果を伝達するデータ順方向分配バスを有し、外部メモリと交信し、対応するトラップレベルを各々有するネストされたトラップをとるため予め定められた数のトラップレベルを支持している、命令発行ユニットによって発行された命令を実行するための処理ユニット中で、命令実行効率を増大するための方法において、

- ー プロセッサ内の投機的命令をトラッキングする段階；
- ー 精確な例外モデルを維持する、ロード及びストア命令を含むメモリ参照命令をトラッキングし攻撃的にスケジューリングする段階；
- ー 精確なアーキテクチャ状態を維持する一方でチェックポイント実行済みの状態を低減させるべく前記処理ユニット内の命令をチェックポイント実行する段階；
- ー 前記実行手段内に発生する例外について例外取り扱いを束縛する段階；
- ー 複数の投機的発行済みの予測された命令の実行結果を同時に監

視する段階；

- ー 浮動小数点例外を検出する段階；
 - ー 投機的にトラップをとりここから復帰する段階；及び
 - ー プロセッサ状態回復を要求する条件を検出した時点で任意の命令境界でプロセッサ内の先行するマシン状態を回復する段階；
- を含んで成る方法。

238. プロセッサ内での投機的命令実行をトラッキングする前記段階には、

- ー データ記憶装置内のデータ構造を規定する段階；
- ー 発行ユニットによって発行された各命令に対して識別タグを割当てする段階；

- ー 割当てられたタグに基づいて各々の発行済み命令に対して、データ構造内に記憶された1つの活動ビットを結びつける段階；
- ー 命令が発行された時点で、データ構造内に活動ビットをセットする段階；及び
- ー 命令がエラー無しで実行を完了した時点でデータ構造内の活動ビットをクリアする段階、

が含まれている、請求項237に記載の方法。

239. 精確な例外モデルを維持する、ロード及びストア命令を含むメモリ参照命令をトラッキングし攻撃的にスケジュールする前記段階には、

- ー 前記プロセッサによる実行のための複数の命令を発行する段階；
- ー 前記発行済みの複数の命令のうちのいずれが投機的に発行された命令であるかを識別する段階；
- ー 前記内部データ記憶装置内に、前記識別された投機的に発行された命令の各々に付随する投機的実行インジケータを記憶する段階

；

- ー 前記発行済み命令のいずれが外部メモリを参照するかを決定する段階；
- ー 前記内部データ記憶装置の中に前記決定されたメモリ参照命令に付随するメモリ参照命令インジケータを記憶する段階；
- ー 前記命令が発行された後前記複数の命令のうちの各々の命令の実行活動状況を監視する段階；
- ー 各々の発行済み命令について実行中に何らかのエラー条件が発生したか否かを確認し、実行中にエラーを経験した各々の命令のためにエラー状況を表示するエラー条件インジケータを生成する段階；
- ー 前記発行済み命令の実行状況をトラッキングする段階；及び
- ー その他の発行されたものの実行されていない命令の実行状況に基づいて、順序的により早期の発行済みの非メモリ参照命令に先立ち、out-of-order実行のために前記決定されたメモリ参照命令のうちの特定の1つの命令をスケジュールする段階であって、この実行状況には、投機的に発行された命令であるものとして

の非メモリ参照命令の識別及び予め定められた実行完了状況を有するものとしての非メモリ参照命令の識別が含まれている、スケジュール段階、が含まれている請求項238 に記載の方法。

240. 前記CPU のための精確なアーキテクチャ状態を維持する一方でチェックポイント実行済み状態を低減させるべく前記プロセッサ内の命令をチェックポイント実行する前記段階には、

- ー 前記CPU 内で実行されたときにアーキテクチャ状態を修正する可能性のある命令を、その発行及び実行前に予め識別する段階；
- ー 予め定められた選択基準に基づいて実行する前にアーキテクチャ状態をチェックポイント実行せずに特別な実行モードにて実行す

るための、前記識別された命令のうちの特定の命令を予め選択する段階；

- ー 実行に先立ち前記予め選択された特定の命令以外の前記識別された命令についてアーキテクチャ状態をチェックポイント実行する段階；及び
- ー 前記特殊モードでの前記命令のうちの特定の命令の実行を含め、前記識別された命令を実行する段階、

が含まれている請求項239 に記載の方法。

241. 前記実行手段内で発生する例外について例外取扱いを束縛する前記段階には、

- ー タイムアウト条件を規定する特定された事象の発生回数について予め定められた閾値を設定し、この閾値を前記CPU 内の第1のデータ記憶装置の中に記憶する段階、
- ー 前記CPU 内のカウンタ内で前記特定された事象の発生を計数し、前記発生回数を前記CPU 内の第2のデータ記憶装置内に1つの計数として記憶する段階；
- ー 前記計数を前記閾値と比較する段階；
- ー 前記計数が前記タイムアウト条件以上である場合に、タイムアウトチェックポイントを形成する段階、

が含まれている、請求項240 に記載の方法。

242. 複数の投機的発行済みの予測された命令の実行結果を同時に監視する前

記段階には、

ー プロセッサ内の前記データ順方向分配全体にわたり前記実行ユニットからの実行結果信号を受理するように結合されたウォッチポイントデータを記憶するための複数のウォッチポイントレジスタをもつウォッチポイントユニットを提供する段階；

ー 前記投機的に発行された予測された命令の各々について、制御

フロー転送方向を左右し投機的発行済み予測命令を投機的に発行する基礎となる1つの条件の予測値を識別する予測条件データ結果を含むウォッチポイントデータを記憶するために1つのウォッチポイントレジスタを割振る段階；

ー 前記データ順方向送りバス上で伝送される前記投機的発行済みの予測命令についての実行結果信号を監視する段階；

ー 前記記憶されたウォッチポイントデータ及び実際の既知の条件データ結果信号及び予め定められた規則を含む前記記憶されたウォッチポイントデータ及び実行結果信号に基づき、予め定められた事象の発生を検出する段階であって、ここで前記実際の既知の条件データ結果信号が、前記投機的発行済みの予測命令の制御フロー転送方向を決定する上で基礎となるべき条件の実際の値を識別しているような、段階；

ー 信号が一致しているかしていないかを決定するべく前記投機的に発行された命令に関して前記データ順方向送りバス上で到着した前記結果信号と前記投機的に発行された予測命令のうちの1つについて前記ウォッチポイントレジスタ内に記憶された前記ウォッチポイントデータを比較する段階であって、一致は前記投機的発行済み予測命令が正しく予測されていたことを表わし、不一致は、前記投機的発行済み予測命令が誤予測されていたことを表わすような、段階；及び

ー 比較の結果、前記予測が誤予測であったことが表示された場合には、前記誤予測に基づいて実行された命令が取消されるように前記中央処理ユニットをより早期の中央処理ユニット状態まで回復させる段階、

が含まれている請求項241 に記載の方法。

243. 浮動小数点例外を検出する前記段階には、

ー 実行のためプログラム制御順で命令を発行する段階であって発行される命令には、浮動小数点及び非浮動小数点命令が含まれている段階；

ー 少なくとも浮動小数点命令が、実行手段によってプログラム制御順外でout-of-order実行され得るように発行済み命令を実行する段階；

ー ・発行済み命令の各々が記憶要素の1つに対応し、各々の記憶要素が浮動小数点命令識別フィールドと浮動小数点トラップタイプフィールドを有する、記憶要素を含むデータ記憶構造を提供する段階

ー ・各々の発行済み命令について対応する発行済み命令が浮動小数点命令であるか否かを表示する対応する記憶要素の浮動小数点命令識別フィールド内へのデータを書込む段階；

ー ・実行中に浮動小数点実行トラップの予め規定された複数のタイプのうちの対応する1つのタイプを結果としてもたらすことになる単数又は複数の浮動小数点実行例外をひき起こす各々の発行された浮動小数点命令について、結果としてもたらされることになる浮動小数点実行トラップの予め定められたタイプのうちの1つを識別する対応する記憶要素の浮動小数点トラップタイプフィールド内へのデータを書込む段階；

ー 実行中実行例外をひき起こさず、プログラム制御順でそれに先行する全ての発行済み命令が退去されてしまっている各々の発行済み命令を退去させる段階；

ー 予め定められた実行例外のうちの第1のものが発行済み命令によってひき起こされた時点で、発行済み命令の実行を続行し、退去され得ない発行済み命令に遭遇するまで発行済み命令を退去させ続けることにより実行トラップ順序付けに着手し、ここで退去され得

ない発行済み命令は、（a）第1の実行例外をひき起こした発行済み命令、及び（b）第1の実行例外をひき起こした発行済み命令よりも早期に発行されたものの第1の実行例外よりも晩期に発生する第2の実行例外をひき起こした発行済み命令のうちの1つである段階；

ー 浮動小数点トラップタイプフィールドをもつ浮動小数点状況レジスタを提供する段階；及び

ー 退去され得ない命令に対応する記憶要素の浮動小数点識別フィールド内のデータが、この退去され得ない命令が浮動小数点命令であることを表示した時点で、退去され得ない命令に対応する記憶要素の浮動小数点トラップタイプフィールド内のデータにより識別された浮動小数点実行トラップのタイプを識別する浮動小数点状況レジスタの浮動小数点トラップタイプフィールドに対するデータを書込む段階；

が含まれている請求項242 に記載の方法。

244. 投機的にトラップをとりこのトラップから復帰する前記段階には、

- ー チェックポイントを形成する段階；
- ー チェックポイントをバックアップする段階；
- ー トラップを取る段階；
- ー トラップから復帰する段階；
- ー トラップが取られる度毎のデータプロセッサの状態を規定する内容をもつレジスタを提供する段階；
- ー トラップレベルより多い数のトラップスタック記憶エントリを有するトラップスタックデータ記憶構造を提供する段階；
- ー トラップレベルのうちの1つにマッピングするため現在利用可能なトラップスタック記憶エントリの現行の利用可能性リストを維

持し、トラップがとられる毎にトラップレベルのうちの対応するものにマッピングするため現在利用可能なトラップスタック記憶エントリのうちの次のものを識別する段階；

- ー とられた各々のトラップについて、現在利用可能なトラップスタック記憶エントリのうちの次のものに対しレジスタの内容を書込む段階；
- ー トラップがとられる毎に、トラップスタック記憶エントリのうちの1つに対する対応するトラップレベルの古いマッピングを、現在利用可能なトラップスタック記憶エントリのうちの次のエントリに対する対応するトラップレベルの現行マッピングと置換することによってトラップスタック記憶エントリのうちの1つに対する各トラップレベルの現行のマッピングを維持する段階、

ー トラップがとられる毎に、古いマッピングにより対応するトラップレベルにマッピングされたトラップスタック記憶エントリを利用不能性リストに付加し、とったトラップをもはや取消しできなくなる毎に、古いマッピングにより対応するトラップレベルにマッピングされたトラップスタック記憶エントリを利用不能性リストから除去することによって、現行マッピングによりトラップレベルの1つに現在マッピングされていないもののトラップレベルの1つにマッピングするためには利用できない各々のトラップスタック記憶エントリを利用不能性リストを維持する段階；

ー 現行の利用可能性リストに対し、利用不能性リストから除去された各々のトラップスタック記憶エントリを付加する段階；

ー チェックポイント記憶エントリを含み、形成された各チェックポイントが対応するチェックポイント記憶エントリをもつような、チェックポイント記憶ユニットを提供する段階；

ー 各々の形成されたチェックポイントについて、対応するチェッ

クポイント記憶エントリ内で、現行のマッピング及び現行の利用可能性リストを記憶する段階；

ー チェックポイントに対する各々のバックアップについて、現行のマッピングを対応するチェックポイント記憶エントリ内に記憶されたマッピングと置換し、現行の利用可能性リストを対応するチェックポイント記憶エントリ内に記憶された利用可能性リストを置換する段階、

が含まれている請求項243 に記載の方法。

245. プロセッサ状態の回復を要求する条件を検出した時点で任意の命令境界でプロセッサ内の先行するマシン状態を回復する前記段階には；

ー 各々の発行済み命令について1つの命令シリアル番号を割振る段階；

ー マシン状態を変更する実行可能な命令の第1の予め定められたセットについてのみ実行に先立って前記プロセッサ内のデータ記憶装置内で1つのチェックポイント中にマシン状態情報を記憶する段階であって、ここでマシン状態を変える実行可能な命令の前記予め定められたセットが、このプロセッサ内で実施された

マシン状態を変化させる実行可能な命令の全てのセットよりも少ないような段階
；

－ 前記障害をひき起こす命令を識別し、多数の例外又は障害が同時に発生した場合には、障害又は例外をひき起こす順序的に最も早期のin-order命令を識別する段階；

－ 前記最も早期の障害発生命令がチェックポイント実行された命令である場合には、前記障害発生命令の実行に先立って記憶された前記マシン状態情報を回復させプロセッサプログラムカウンタを前記障害発生命令の命令シリアル番号まで減分させて前記先行するマ

シン状態を回復する段階；

－ マシン状態情報が記憶されてきた前記予め定められた実行可能な命令の1つが順序的に前記障害発生命令と最後に発行された命令の間に置かれている場合には、(i) まず最初に、順序的に前記障害発生命令の後で前記障害発生命令に最も近いチェックポイントまで前記プロセッサをバックアップし、(ii) 第2にレジスタ資源を更新することによって前記障害命令の実行の直前に存在した状態までプロセッサの状態を回復するように前記プロセッサをバックステップさせ、(iii) 前記障害発生命令の命令シリアル番号までプロセッサプログラムカウンタを減分させる段階；

が含まれている請求項244 に記載の方法。

【発明の詳細な説明】

特殊機能を提供する高性能投機的実行プロセッサの構造及び方法

関連出願

この出願は発明者Gene W.Shen et al.によって1995年3月3日に出願された、米国特許出願代理人整理番号第A-60625-1/JAS、米国特許出願第08/398,299号 “PROCESSOR STRUCTURE AND METHOD FOR TRACKING INSTRUCTION STATUS TO MAINTAIN PRECISE STATE” の継続であり；該継続出願は同じく発明者Gene W.Shen et al.によって1995年2月14日に出願された、米国特許出願代理人整理番号第A-60625/JAS、米国特許出願第08/390,885号 “PROCESSOR STRUCTURE AND METHOD FOR TRACKING INSTRUCTION STATUS TO MAINTAIN PRECISE STATE” の継続である。

発明者Takeshi Kitaharaによって1995年2月14日に出願された、米国特許出願代理人整理番号第1706号、米国特許出願第08/388,602号 “INSTRUCTION FLOW CONTROL CIRCUIT FOR SUPERSCALER MICROPROCESSOR” ；発明者Michael Simone及びMichael Shebanowによって1995年2月14日に出願された、米国特許出願代理人整理番号第1693号、米国特許出願第08/388,389号 “ADDRESSING METHOD FOR EXECUTING LOAD INSTRUCTIONS OUT OF ORDER WITH RESPECT TO STORE INSTRUCTIONS” ；発明者DeForest Tovey,Michael Shebanow,John Gmuender によって1995年2月14日に出願された、米国特許出願代理人整理番号1707号、米国出願第08/388,606号 “METHOD AND APPARATUS FOR EFFICIENTLY WRITING RESULTS TO RENAMED REGISTERS” ；及び発明者DeForest Tovey,Michael Shebanow,John Gmuender によって1995年2月14日に出願された、米国特許出願代理人整理番号第1741号、米国出願第388,364号 “METHOD AND APPARATUS FOR COORDINATING THE USE OF PHYSICAL REGISTERS IN A MICROPROCESSOR” はいずれも参考のため本願明細書にその内容を引用する。

発明者Gene W.Shen,John Szeto,Niteen A.Patkar及びMichael C.Shebanowによって1995年2月14日に出願された、米国特許出願第08/390,885号 “PROCESSOR STRUCTURE AND METHOD FOR TRACKING INSTRUCTION STATUS TO MAINTAIN PRECISE STATE” ；発明者Gene W.Shen,John Szeto,Niteen A.Patkar及びMichael C.Shebanow

owによって1995年3月3日に出願された米国特許出願第08/398,299号“PROCESSOR STRUCTURE AND METHOD FOR TRACKING INSTRUCTION STATUS TO MAINTAIN PRECISE STATE”；発明者Chih-Wei David Chang,Kioumars Dawallu,Joel F.Boney,Min g-Ying Li,及びJen-Hong Charles Chenによって1995年3月3日に出願された、米国特許出願第08/397,810号“PARALLEL ACCESS MICRO-TLB TO SPEED UP ADDRES S TRANSLATION”；発明者Leon Kuo-Liang Peng,Yolin Lih,及びChih-Wei David Changによって1995年3月3日に出願された、米国特許出願第08/397,809号“LOO KASIDE BUFFER FOR ADDRESS TRANSLATION IN A COMPUTER SYSTEM”発明者Michael C.Shebanow,Gene W.Shen,Ravi Swami,及びNiteen Patkar によって1995年3月 3日に出願された米国特許出願第08/397,893号“RECLAMATION OF PROCESSOR RES OURCES IN A DATA PROCESSOR”；Michael C.Shebanow,John Gmuender,Michael A .Simone,John R.F.S Szeto,Takumi Maruyama, 及びDeForest W.Toveyによって19 95年3月3日に出願された米国特許出願第08/397,891号“METHOD AND APPARATUS FOR SELECTING INSTRUCTIONS FROM ONES READY TO EXECUTE”；Shalesh Thusoo ,Farnad Sajjadian,Jaspal Kohli,及びNiteen Patcar によって1995年3月3日 に出願された米国特許出願第08/397,911号“HARDWARE SUPPORT FOR FAST SOFTWA RE EMULATION OF UNIMPLEMENTED INSTRUCTIONS”，Akira Katsuno,Sunil Savkar , 及びMichael C.Shebnow によって1995年3月3日に出願された米国特許出願第 08/398,284号“METHOD AND APPARATUS FOR ACCELERATING CONTROL TRANSFER RET URNS”，Akira Katsuno,Niteen A.Patcar,Sunil Savkar, 及びMichael C.Shebno w によって1995年3月3日に出願された米国特許出願第08/398,066号“METHODS FOR UPDATING FETCH PROGRAM COUNTER”；Sunil Savkarによって1995年3月3日 に出願された米国特許出願第08/398,151号“METHOD AND APPARATUS FOR RAPID E XECUTION OF CONTROL TRANSFER INSTRUCTIONS”；Chih-Wei David Chang,Joel F redrick Boney, 及びJaspal Kohliによって1995年3月3日に出願された米国特 許第08/397,910号“METHOD AND APPARATUS FOR PRIORITIZING AND HANDLING ERR ORS IN A COMPUTER SYSTEM”；Michael Simoneによって1995年3月3日に出願さ れた米国特許出願第08/397,800号“METHOD AND APPARATUS FOR GENERATING A ZE

RO BIT STATUS FLAG IN A MICROPROCESSOR”；及びChien Chen及びYizu Lu によって1995年 3 月 3 日に出願された米国特許出願第08/397,912号 “ECC PROTECTED MEMORY ORGANIZATION WITH PIPELINED READ-MODIFY-WRITE ACCESS” もそれぞれ参考のため本願明細書中に引用した。

発明者Sunil Savkar,Michael C.Shebanow,Gene W.Shen,及びFarnad Sajjadian によって1995年 6 月 1 日に出願された米国特許出願第 号 “METHOD AND APPARATUS FOR ROTATING ACTIVE INSTRUCTIONS IN A PARALLEL DATA PROCESSOR”；及び発明者Sunil Sankar,Michael C.Shebanow,Gene W.Shen,及びFarnad Sajjadianによって1995年 6 月 1 日に出願された米国特許出願第 号 “PROGRAMMABLE INSTRUCTION TRAP SYSTEM AND METHOD” もそれぞれ参考のため本願明細書中に引用した。

発明者Gene W.Shen,John Szeto,Niteen A.Patkar, 及びMichael C.Shebanowによって1995年 6 月 7 日に出願された代理人整理番号第 A -60624/JAS、米国特許出願第08/487,801号 “PROCESSOR STRUCTURE AND METHOD FOR TRACKING INSTRUCTION STATUS TO MAINTAIN PRECISE STATE”；発明者Gene W.Shen,John Szeto,Niteen A.Patkar,Michael C.Shebanow,及びMichael A.Simoneによって1995年 6 月 7 日に出願された、代理人整理番号第 A -60622/JAS、米国特許出願第08/478,025号 “PROCESSOR STRUCTURE AND METHOD FOR AGGRESSIVELY SCHEDULING LONG LATENCY INSTRUCTIONS INCLUDING LOAD/STORE INSTRUCTIONS WHILE MAINTAINING PRECISE STATE”；発明者Gene W.Shen,John Szeto,Niteen A.Patkar, 及びMichael C.Shebanowによって1995年 6 月 7 日に出願された、代理人整理番号第 A -60623/JAS、米国特許出願第08/483,958号 “PROCESSOR AND METHOD FOR MAINTAINING AND RESTORING PRECISE STATE AT ANY INSTRUCTION BOUNDARY”；発明者Gene W.Shen,John Szeto,Niteen A.Patkar, 及びMichael C.Shebanowによって1995年 6 月 7 日に出願された、代理人整理番号第 A -60625-2/JAS、米国特許出願第08/467,419号 “PROCESSOR STRUCTURE AND METHOD FOR CHECKPOINTING INSTRUCTIONS TO MAINTAIN PRECISE STATE”；発明者Gene W.Shen,John Szeto,Niteen A.Patkar, 及びMichael C.Shebanowによって1995年 6 月 7 日に出願された、代理人整理番号第 A -60

647/JAS、米国特許出願第08/473,223号“PROCESSOR STRUCTURE AND METHOD FOR A TIME-OUT CHECKPOINT”；発明者Gene W.Shen, John Szeto, 及びMichael C.Shebanowによって1995年6月7日に出願された、代理人整理番号第A-60648/JAS、米国特許出願第08/484,795号“PROCESSOR STRUCTURE AND METHOD FOR TRACKING FLOATING-POINT EXCEPTIONS”；発明者Hideki Osone及びMichael C.Shebanowによって1995年6月7日に出願された、代

理人整理番号第A-60649/JAS、米国特許出願第08/472,394号“PROCESSOR STRUCTURE AND METHOD FOR RENAMABLE TRAP-STACK”；及び発明者Gene W.Shen, Michael C.Shebanow, Hideki Osone, 及びTakumi Maruyama によって1995年6月7日に出願された、代理人整理番号第A-60682/JAS、米国特許出願第08/482,073号“PROCESSOR STRUCTURE AND METHOD FOR WATCHPOINT FOR PLURAL SIMULTANEOUS UNRESOLVED BRANCH EVALUATION”もそれぞれ参考のため本願明細書に引用した。

技術分野

本発明は投機的追越し(out-of-order)実行プロセッサにおいて精確な状態(precise state)を維持しながら、プロセッサの性能を高める装置、システム、及び方法に係わる。

発明の背景

プロセッサにおいて、制御流れ命令(例えば分岐命令)、メモリのトランザクションに起因する待ち時間(latency)、及びマルチサイクル演算を必要とする命令は、パイプラインに“バブル”を導入する原因となるため、プロセッサが高い命令実行帯域幅を維持するのを妨げることが多い。投機的追越し命令実行を行うことによって性能を高めることができる。従来、1つの命令からの中間結果を後続の命令に利用できないとき、プロセッサは実行を中断するか、または中間結果が得られるまで“機能停止する”。

2つの技術、即ち、投機的実行と追越し実行によって、この新しいプロセッサは高い実行帯域幅(execution bandwidth)を維持することができる。投機的実行は、先行の処理ステップから該当分岐を選択するための情報がないままに1つの分岐に遭遇した場合、予測

し、この予測に基づいて命令のディスパッチ及び実行を行う公知の技術である。もし予測が誤りであることが後刻判明したら、分岐誤予測回収によって誤予測された命令シーケンスを取消さねばならない。投機的実行によってプロセッサは命令の発行及び実行を継続することができる。公知の予測スキームは性能を高めるため、誤予測実行の頻度を極力少なくする。しかし、投機的マシンにおいて、適正状態の維持は煩雑であり、好ましくないオーバヘッドを伴う。追越し実行はメモリ及びマルチサイクル命令待ち時間に留意しない技術である。追越し実行を行うプロセッサにおいては、命令の順序をダイナミックに変更し、順次プログラムとは異なる順序で命令を実行することによって利用可能な命令レベル並列性を見つける。

“精確な例外”モデルは例外条件のソフトウェア解像度を単純化する重要な要因ではあるが、投機的追越し実行を行うマシンにおいて、精確な例外を維持するのは煩雑な操作である。マシンの状態は一般にプロセッサに特異であり、構造上の状態はすべての制御／状態レジスタ、データ・レジスタ、アドレス・レジスタ及びすべての補助メモリの状態を含む。例えばSPARC-V9 Architecture Manualのpp.29 - 30にはSPARC-V9制御／状態レジスタが記載されており、ソフトウェア・プログラマーのよく知るところである。マシンの状態はプロセッサに特異な、かつマシンの状態に関する他のすべてを含む構造上の状態のスーパー・セットである。不正命令(faulting instruction)とは例外を発生させる命令である。例外とはプロセッサにその動作を停止させ、処理再開の前にこの例外の原因となった状況を究明するように指示する状況または条件である。例外はエラーとは限らず、例えば、割込みをも含む。実行トラップは、例外から生ずる可能性がある。精確な例外モデルを作成するプロセッサにおいては、不正または例外は構造上の状態を変更しない。不正命令に

先立つすべての命令に関しては構造上の状態がすでに変更されているが、不正命令後の命令に関しては構造上の状態は変更されていない。正確な例外モデルがなければ、ソフトウェアが不正命令を識別し、次いで、不正命令を再試行するか、または不正命令をバイパスして次の命令を実行するための再始動点(restart poi

nt)を算出しなければならない。

短パイプライン単一発行マシンにおいて精確な状態を維持する技術は公知である。一般に、短パイプラインマシンは命令の取出し、発行、実行、及び状態を変更されるライトバック段階を含む約4または5段階以下の段階を有する。単一発行方式によれば、パイプライン段階におけるどの命令をフラッシュすべきかを考慮せずにパイプラインをクリアすればよいから、例外または誤予測の場合に回復が簡単になる。これらの公知技術においては、例外が発生した場合、構造上の状態を変更する前に検出される。例外が検出されると、パイプラインから命令がフラッシュされ、構造上の状態が変更されないように、構造上の状態ヘデータ、状態または結果がライトバックされるのを意図的に阻止する。

投機的追越しスーパスカラ（多重発行）方式では単一発行マシンの場合と比較して、精確な状態を維持することがはるかに困難である。このような投機的追越しマシンの場合には、エラーを発生させる命令が投機的に実行される可能性があり、エラーの場所のあとに現われる構造上の状態の変更を取消す方法及び構造を設ける必要がある。また、例外は多くの場合、プログラムの順序とは異なる順序で検出される。従って、追越しプロセッサは例外を解読し、どの命令を完了（及び構造上の状態を変更）させるべきか、どの命令を取消すべきかを判断できねばならない。

図1は、投機的プロセッサにおいて精確な状態を維持するために

再順序付けバッファを利用する公知のアプローチを示す。再順序付けバッファは、（結果が得られる）命令実行完了時から実行完了の結果としてマシンの状態が変更されるまでの間に一時遅延を有効に導入する先入れ／先出しスタックによって実現される。メモリへの誤予測分岐命令、実行例外、または同様の状態がライトバックされるのを阻止することによって精確な状態を維持する。誤予測に基づく投機的実行を取消すために状態を復旧するのではなく、命令が投機の域を出ない間、状態の変更を阻止することによって、精確な状態を維持するのである。

図2は、命令を実行する前に状態を“チェックポイント”に記憶させ、あとで、記憶されているチェックポイント情報から状態を復旧することにより、投機的

プロセッサにおいて精確な状態を維持する公知のアプローチを示す。公知のチェックポインティングでは、マシンの状態変更を伴う可能性がある、投機的に実行された命令を残らずチェックポイントする。各プロセッサはマシンの状態を決定する一組の状態パラメータ、例えば、すべての制御／状態レジスタ、データレジスタの値などを有する。先のマシン状態を復旧するため、状態を決定するすべてのパラメータを記憶させ、必要に応じて復旧できるようにしなければならない。公知のチェックポインティング技術では、状態を決定するパラメータのいずれか1つでも変更する可能性がある命令に関してすべての状態決定パラメータを記憶させるのが典型的である。チェックポインティングの対象となるすべての命令について、公知のチェックポインティングでは、実行寸前の特定の命令によって変更されそうな状態だけでなく、チェックポインティングの対象である命令のいずれか1つによって変更される可能性のあるすべての状態情報を記憶させる。例えば、マシンの状態を決定するのに100個の状態パラメータを必要とするマシン

の場合、もし命令“X”の実行が1つだけの制御／状態レジスタを変更する可能性があるなら、この命令を実行するには、この命令によって変更されると考えられるパラメータだけでなく、100個の状態パラメータを記憶させる必要がある。

図3は、一連の命令の構造及び内容と公知のチェックポイントとを模式的に示す説明図であり、特定のマシンまたはCPUのための実際のチェックポイント・データを示すものではない。公知のチェックポイントはサイズが一定であるから、各チェックポイントは特定の命令によって実際に変更される状態よりも広くなければならない。公知のチェックポインティングを利用する回復としては、例えば、不正命令または誤予測に基づいて投機的に実行された命令の直前に記憶されているチェックポイントを利用しての復旧、チェックポイントされている命令の直後にまでプログラム・カウンタをバックアップすること、及びチェックポイントされている命令からの命令再実行などがある。

追越し実行を管理し、状況によっては適正状態を維持する手段として、チェックポインティング、再順序付けバッファ、履歴バッファ、及びフューチャファイルの使用がすでに開示されている。任意の命令境界においてではなく、チェッ

クポイント境界においてマシン状態の復旧を可能にする慣用のチェックポイントイングはHwu 及びPattが論述している（第14回コンピュータ・アーキテクチャ年次シンポジウム（1987年6月）における会報pp.18 -26に発表されたW.Hwu 及びY.N.Pattの“Checkpoint Repair for High Performance Out-of-Order Execution Machines”）。パイプラインRISCプロセッサにおける精確な割込みを作成する方法はWang及びEmmettによって報告されている（“Implementing Precise Interruptions in Pipelined RISC Processors”, IEEE Micro,1993年8月、pp.36 -43）。同じくパイプライン・プロセッサにおける精確な割込み作成はSmith 及びPleszkunによっても報告されている（第12回コンピュータ・アーキテクチャ年次国際シンポジウム会報（1985年6月）、pp. 36-44におけるJ.E.Smith 及びA.R.Pleszkunの“Implementation of Precise Interrupts in Pipelined Processors”）。高性能スーパスカラ・マイクロプロセッサの設計に関する従来技術の概要はMike Johnsonが記述している（M.Johnson[a.k.a.William Johnson], Superscaler Microprocessor Design,Prentice-Hall,Inc.,Englewood Cliffs,New Jersey 07632,1991,ISBN 0-13-875634-1）。これらの参考文献の内容は参考のため本願明細書中に引用した。

これらの技術のうち、少なくともいくつかは性能を向上させるが、投機の可能性を制限し、大ざっぱなマシン状態は回復できても命令レベルのマシン状態回復は不可能であり、完全に満足できるものではない。公知の再順序付けバッファ技術は、その良さが投機の可能性と正比例するから、投機の可能性に限界がある。即ち、取消しの対象となりそうなすべての命令に関して、命令実行の結果を再順序付けバッファに記憶させねばならない。例えば、もしマシンが64個の未決定かつ投機的命令を可能にするなら、再順序付けバッファは結果を記憶するための少なくとも64個の記憶場所を含まねばならない。公知のチェックポイントイングにおいては、チェックポイントされるチェックポイント数はマシン中の未定命令数よりも少ないのが普通であるが、各チェックポイントに記憶されるデータ量は極めて大きい（各時点におけるマシンの全体的な状態）。チェックポイント記憶条件はプロセッサのチップ基板面積に負担を課する。理想的には、精確な状態を維

持するためのスキームは比較的多い同時的に存在する投機的命令に対して線形または低位（例えば対数）関

係にスケールできることが望ましい。再順序付けバッファの入口(entry)記憶域はデータを記憶するのに十分な幅を持たねばならず、これらの技術はまたすべての入口までの連想検索を必要とする。これは、非常に大きな再順序付けバッファにとって困難である。

さらに、精確な状態を回復するための従来の方法は一部の環境で不完全である。たとえば、投機的分岐の結果として実行される命令が外部「ダム」デバイスの状態を修正する場合、状態の回復が通常、外部デバイス修正ポイント前のチェックポイント境界での状態回復を伴うため、外部デバイスを修正する命令を含む非不正命令のフォワード再実行を伴う場合である。このような場合、不正命令の再実行だけでは、外部デバイスの状態変化の効果を元通りにせず、状態回復ポイントと不正命令との間で非不正命令を再実行することは、さらに問題を生じる。

従来のチェックポイントニングでは、命令ストリームの限られた数のポイントでマシン状態をセーブし、チェックポイントされたアーキテクチャル状態を復旧して、分岐誤予測または実行エラーから回復する。従来のチェックポイントニングは命令を1つ1つチェックポイントしない。従来のチェックポイントニングは、チェックポイント境界、すなわち、チェックポイントを確立した命令でのみマシン状態を回復できる。フォールトまたは実行エラーが発生すると、チェックポイントされた状態は、周知の技術によって回復され、これによって、チェックポイント後の不正命令を含むすべての命令を事実上「元通り」にする。そして命令は、たとえば単一発行モードで、チェックポイントされた命令から順次前向きに再実行され、不正命令に到達する。

この従来のチェックポイントニング技術では投機の実行が可能だが、多くの点で不利益である。たとえば、これは堅牢な例外回復と

ならない。間欠的エラーでは、例外作成命令前のチェックポイントへの従来のマシンバックアップと、チェックポイント後の命令の再実行は決定的な行動となら

ない場合があり、再実行された命令の誤って変更された状態を伝播することによって、マシン状態の混乱を増すことがある。従来のチェックポインティングとマシンバックアップ手順は、命令の再実行を最小限にしようとするものではない。さらに、ハードウェアの故障やマシンのタイムアウトなど災害的なマシンエラーは、回復したチェックポイント済み命令と例外を生じる命令との間のすべての命令が再実行されると、プロセッサをデッドロックさせることがある。

従来の「再順序づけバッファ」とは、予め定義された大きさの先入れ先出し（F I F O）メモリ構造の再順序づけバッファで投機的命令を管理する。命令が実行を完了したら、実行によるデータ値の結果が再順序付けバッファに書き込まれ、バッファを移動して上位に現れる時、データ値が再順序づけバッファからレジスタファイルに書き込まれる。再順序づけバッファの大きさは、命令実行の完了とアーキテクチャル状態の恒久的修正との間の遅延を効果的に定義する。一旦データ値がレジスタに書き込まれると、これらを元通りにすることはできない。再順序付けバッファの記述項（エントリ）をレジスタファイルの記述項と関連づける手順である「連想探索」について、M.Johnson が Superscalar Microprocessor Design の 4 0 頁以降で説明している。

再順序付けバッファスキームには少なくとも 3 つの限界がある。1 つ目は、従来の再順序づけバッファスキームでは、命令実行の結果のみが再順序付けバッファにセーブされ、プログラムカウンタ（P C）値はセーブされない。そのため、再順序づけバッファを使った分岐誤予測回復には、P C 再構築、命令フェッチ、命令発行の追

加ステップが必要となる。その結果、再順序付けバッファを使った分岐誤予測からの従来の回復は遅れる。

第 2 に、再順序づけバッファは一般に限られた命令ミックスの投機的実行しかできない。たとえば、命令発行段階中に検出されたトラップは（発行トラップ）、一般に制御レジスタアップデートを伴うため、再順序付けバッファを使って投機的に実行されないことがある。再順序付けバッファ技術は、制御レジスタアップデートを伴う命令の投機的実行をサポートしない。一定の命令セットアーキテ

クチャで発行トラップを投機的に入力できないこと（例：Sun Microsystems SPA RCアーキテクチャの「スピル／フィル」トラップ）は、大きな性能上の限界となる。

第3に、再順序付けバッファの大きさは一般に、プロセッサに許された未定命令数の直接線形関数である。たとえば、64個の未定命令の可能なプロセッサでは、64個の記述項を持つ再順序付けバッファが必要となる。プロセッサ内に多数のバッファレジスタを割り当てることは禁止されることがあり、特に大型のアクティブ命令ウィンドウ、すなわち比較的多数の同時未定命令によって、命令レベルの並列性の抽出が改善されるデータフロープロセッサに当てはまる。データフロープロセッサとは、命令実行の順序が、従来の非データフロープロセッサのようにプログラムカウンタのインクリメントによるのではなく、オペランドやデータ利用可能性によって決定するプロセッサである。

フューチャファイルは、再順序付けバッファの連想探索問題を避けるための再順序づけバッファ技術の修正である。フューチャファイルについては、M. Johnson の Superscalar Microprocessor Design の 94 頁から 95 頁に説明されている。履歴バッファは、順序無視 (out-of-order) 完了のパイプライン・スカラー・プロセッサで

精確な割り込みを実行するため提案された方法と構造で、M. Johnson の Superscalar Microprocessor Design の 91 頁から 92 頁に説明されている。

本書で全体を参照することにより明示的に組み入れる D. Weaver と T. Germond による The SPARC Architecture Manual N Version 9、Englewood Cliffs (1994 年) では、順序無視投機的実行プロセッサの特定のタイプについて説明している。SPARC V9 アーキテクチャには、浮動小数点状態レジスタ (FSR) が必要である。FSR には、FSR__accrued__exception (FSR. aexc) フィールド、FSR__current__exception (FSR. cexc) フィールド、FSR__floatingpoint__trap__type (FSR. ftt) フィールドの 3 つのフィールドがあり、浮動小数点例外が発生すると更新されて、トラップハンドリングルーチンが、浮動小数点例外に

よるトラップに対処するため利用する。これらフィールドのアップデートは、命令がプログラム順序とは異なる順序で実行、完了するため、順序無視実行プロセッサでは難しい。これらフィールドは、命令がプログラム順序で発行、実行されたかのように更新されるか、更新されたように見えなければならないため、これらの例外を追跡してF S Rレジスタを正しく更新するためには、装置および対応する方法が必要である。

命令を投機的に実行できるデータプロセッサでは、分岐方向（取るか取らないか）、あるいは分岐アドレス（目標アドレスまたは分岐命令の次のアドレス）は、解決する前に予測可能である。後に、これら予測が誤っていることがわかれば、プロセッサは以前の状態にバックアップして、正しい分岐ストリームで命令実行を再スタートする。但し、市販のスーパースカラプロセッサの大半は、分岐予測

が正しいかどうかをチェックするには、1サイクルあたり1つの分岐のみ評価可能である。しかし、多重予測分岐は1つのサイクルで評価できることが多い。そのため、他の場合では実行可能な分岐評価が遅延することになる。分岐評価の遅延は、プロセッサの性能に大きな影響を与える。

さらに、従来の投機的実行プロセッサでは、トラップが生じると、プロセッサは予測されたすべての分岐が解決してトラップが真実であり投機的ではないことを確かめるまで待たなければならない。トラップが真実であるかどうかをプロセッサが確かめる最も簡単な方法は、トラップを取る前にプロセッサを同期させることである（すなわち、トラップ条件発生前に発行されたすべての命令を実行、完了する）。但し、頻繁に発生するトラップにこれを行うと、プロセッサの性能を落とす。スピル／フィル発行トラップとソフトウェアトラップ（T c c 命令）がしばしば発生するSPARC-V9アーキテクチャでは特にそれが当てはまる。プロセッサの性能を上げるためにはこの問題を解決しなければならない。

発明の概要

先行技術の前記問題は、多数の特殊プロセッサ機能および能力を提供する構造および方法を含む高性能プロセッサに関する発明で対処される。これら構造および方法には、（1）精確状態を維持しながらロード／ストア命令を含む長待ち時

間 (latency) 命令の積極的なスケジューリング；あらゆる命令境界での精確状態の維持と回復、(3) 精確状態を維持するための命令状態の追跡、(4) 精確状態を維持するための命令のチェックポイントニング、(5) タイムアウトチェックポイントの作成、維持および利用、(6) 浮動小数点例外の追跡、(7) リネーム可能なトラップスタックの作成、維

持および利用、(8) 複数の同時未解決分岐評価のためのウォッチポイントの作成、維持および利用、(9) 精確状態維持のための命令状態追跡、(10) 精確状態を維持しながらのプロセッサスループット改善のための構造および方法、を限定せずに含む。他の構造機能および能力は下記の開示および添付の図面と請求の範囲で説明される。

前記問題は、本発明の方法および構造の一側面によって解決されるが、これは、発行時に各命令に独自の識別タグを割り当て、そのタグを最初のアクティブな命令データ構造で記憶場所に関連させ、各命令の命令アクティビティ状態の変化に対応して記憶場所に記憶されたデータを更新し、命令アクティビティ状態に対応して移動する記憶場所に複数のポインタを維持することによって、精確状態の追跡と維持を行うものである。状態情報には、命令発行時に設定され、エラーなしに実行が完了するとクリアされる1つのアクティブビットが含まれる。最後に発行された命令を指すポインタ（発行した命令のポインタ）、エラーなしに完了し、順序的に早い命令すべてがエラーなしに完了した命令を指すポインタ（最後にコミットした命令のポインタ）、割り当てられたプロセッサリソースを取り戻した最後の命令（取り戻した命令のポインタ）が設定される。これら3つのポインタは、データ構造の1つの命令に関連する各場所のアクティブビットの比較および所定の規則に基づいて、データ構造に沿って後に発行された命令に向かって前進する。命令の例外またはエラー条件は、アクティブビットの変更を妨げるので、ポインタの移動が制御され、これら条件下で妨げられる。

前記問題は、本発明の方法および構造の別の側面によって解決されるが、これは、ロード/ストア命令を含む、外部メモリ参照命令などの長待ち時間命令を、投機的順序無視実行プロセッサの短待ち

時間命令の前に追跡および積極的にスケジューリングしながら、メモリ参照命令を他の命令と区別し、どの命令の予測が終わって投機的結果を有するかを識別し、投機的分岐誤予測や実行例外の懸念なしに実行可能な実行命令を参照するメモリのみスケジューリングすることによって、精確状態を維持する。このようにスケジューリングできる命令は、発行時に各命令に独自の識別タグを割り当て、そのタグをデータ構造の記憶場所に関連させ、各命令の命令アクティビティ状態の変化に対応してデータ構造の記憶場所に記憶されたデータを更新し、命令アクティビティ状態に対応して移動する記憶場所に複数のポインタを維持することによって、追跡する。状態情報には、命令発行時に設定され、エラーなしに実行が完了するとクリアされる1つのアクティブ命令インジケータが含まれる。最後に発行された命令、エラーなしに完了し、順序的に早い命令すべてがエラーなしに完了した命令（最後にコミットした命令）、割り当てられたプロセッサリソースを取り戻した最後の命令（取り戻した命令）を指すポインタが設定される。さらに、データ構造の第2の記憶場所にタグに関連させ、メモリ参照命令のような長待ち時間命令について命令アクティビティ状態変化に応じて第2記憶場所に記憶されたデータを更新し、長待ち時間（メモリ参照等）命令アクティビティ状態に対応して移動する記憶場所に複数のポインタを維持する。状態情報には、長待ち時間命令が発行された時点でクリアされるが、他の命令タイプすべてについて設定され、命令実行がエラーなしに完了した時にクリアされる長待ち時間命令タイプが含まれる。最も早く予測された分岐命令を指す第1ポインタ（予測分岐ポインタ）がデータ構造に設定される。エラーなしに完了し、順序的に前のすべての命令がエラーなしに完了し、アクティブな長待ち時間命令を越えて前進する最後の命令を指す第2ポインタが設定される。

発行、コミット、リタイア、予測された分岐と、長待ち時間ポインタは、各場所のアクティブビットとメモリビットの比較および所定の規則に基づき、第1および第2環状データ構造に沿って後に発行された命令に向かって前進する。例外または命令のエラー条件は、アクティブビットとメモリビットの変更を妨げるため、これら条件下ではポインタの移動は妨げられる。1以上のデータ構造を設けて

アクティビティ状態と命令タイプデータを記憶することができ、インジケータはデータ構造に記憶されるセットまたはクリアされたビットでもよい。

前記問題は、本発明の方法および構造の別の側面によって解決されるが、これは、分岐命令や、プログラムカウンタ不連続性を作ったり、制御レジスタ値を修正する副作用のある命令を含む命令タイプの所定セットにのみにチェックポイントを作成し、プログラムカウンタの不連続性を作る各命令にチェックポイントを作成するか、プロセッサを同期させて状態を修正することのある命令を順序通りに実行して、プロセッサを例外から回復するための手段および方法を提供することで、投機的順序無視実行プロセッサの命令境界で精確なマシン状態を回復する方法を提供する。例外条件からの回復には、実行例外が発生した時の命令一連番号、またはサイクル中に1つ以上の実行例外が発生した時の最も早い命令一連番号の決定および記憶と、実行例外後に最も近い前回チェックポイントした命令にプロセッサをバックアップし、チェックポイントが現在発行されている命令と不正命令との間にある場合、チェックポイントした情報からプロセッサ状態を回復し、レジスタリソースを更新し、チェックポイントした命令があれば、ここから最も早い不正命令の直前のポイントまでのバックステップ量でプロセッサのプログラムカウンタをデクリメントすることによるプロセッサのバックステップが含ま

まれる。今後の改良では、従来のチェックポイントニングに必要な多量の記憶装置は、レジスタデータそのものではなく、ロジカルおよびフィジカルレジスタリネームマップをチェックポイントニングすることによって減少する。

前記問題は、本発明の方法および構造の別の側面によって解決されるが、これは、プロセッサにチェックポイントニング命令を設けてチェックポイントした状態を減らす一方、命令の発行および実行前にCPUで実行された時アーキテクチャル状態を修正するような命令を予め識別し、特定の命令のチェックポイントに必要な状態記憶量と典型的命令ストリームで命令が発生する頻度を含む所定の選択基準に基づき、実行前に特定の命令のアーキテクチャル状態をチェックポイントニングすることなしに特別実行モードで実行するため識別された命令の内の1つを予め選ぶことによって、プロセッサの精確状態を維持する。そして、チ

チェックポイントは識別した命令についてのみ形成し、状態を修正するような他の命令はチェックポイントせず、特別な同期モードで実行し、例外が発生した時に、実行結果をプロセッサ状態に書き戻す前にその例外に対処できるようにする。本発明の方法および構造の1実施例では、この基準は、修正可能な状態の種類と、実行中に命令によって修正されることのある修正可能状態の量からなり、おそらくは所定の名目命令ストリームで命令の発生する頻度が含まれる。低推定統計頻度を持って現れ、比較的大きなチェックポイント記憶量の必要な命令を同期モードの実行に選択し、チェックポイントしないことが有利である。本発明の構造および方法の1実施例では、プロセッサの同期においては、実行前に命令ストリームでのマシン同期を必要とする同期命令として命令を識別し、未解決の発行済み命令をすべてコミットし、リタイアして、エラーなしに実行を完了して実行結果を状態に書

き戻すまで同期命令実行を遅延させ、マシン同期の必要とする各命令を逐次順序通り実行し、状態へのライトバック前に各同期命令実行から生じる例外条件を識別し、同期命令実行中に生じることのある例外条件に対処し、その後により実行結果をマシン状態に書き戻すようにして、精確状態を同期命令のチェックポイントニングなしに維持できるようにする。

前記問題は、本発明の方法および構造の別の側面によって解決されるが、これは、デコードした命令属性にのみ基づいて現在のプロセッサ状態を記憶するためのチェックポイントを形成するのではなく、発行した命令数、経過したクロックサイクル数、および最後にチェックポイントを形成してからの間隔に基づいてチェックポイントを形成する。本発明のチェックポイントニングに基づくタイムアウト条件ベースのチェックポイントニングは、命令の最大数をチェックポイント境界以内に限定するため、例外条件からの回復期間を抑制する。命令ウィンドウのサイズがチェックポイント境界内の命令最大数より大きい限り、例外発生時、プロセッサは命令デコードベースのチェックポイント技術より早くチェックポイントした状態を復旧することができ、プロセッサの命令ウィンドウサイズへの状態回復依存をなくす。本発明のタイムアウトチェックポイント形成は、従来のチ

チェックポイントを実行する構造および方法、または本発明のロジカルおよびフィジカルなレジスタリネームマップチェックポインティング技術に用いることができる。本発明のタイムアウトチェックポイント形成構造および方法は、従来のプロセッサバックアップ技術、およびプロセッサバックアップとバックステッピングを含む本発明のバックトラッキング技術と共に用いることができる。

前記問題は、本発明の構造および方法の別の側面によって解決さ

れるが、これは、発行ユニット、実行手段、浮動小数点例外ユニット、精確状態ユニット、浮動小数点状態レジスタおよび書き込み手段からなるプログラム制御順序無視実行データプロセッサを提供する。発行ユニットは実行のプログラム制御順序で命令を発行する。発行された命令には浮動小数点命令と非浮動小数点命令が含まれる。実行手段は、少なくとも浮動小数点命令が実行手段によってプログラム制御順序を無視して実行されるよう、発行された命令を実行する。浮動小数点実行ユニットには、記憶要素をはじめとするデータ記憶構造が含まれる。発行された各命令は記憶要素の1つに対応する。各記憶要素は、浮動小数点命令識別フィールドと、浮動小数点トラップタイプフィールドを有する。浮動小数点例外ユニットにも、対応する記憶要素の浮動小数点命令識別フィールドに、発行された各命令に関してデータを書き込み、対応する発行済み命令が浮動小数点命令か否かを示すための第1ロジックが含まれる。これはさらに、実行中に1つ以上の浮動小数点実行例外を生じて、浮動小数点実行トラップの予め定義された複数の種類の対応する1つとなる発行済み浮動小数点命令について、対応する記憶要素の浮動小数点トラップタイプフィールドにデータを書き込み、結果として生じる浮動小数点実行トラップの予め定義された種類の1つを識別する第2のロジックが含まれる。精確状態手段は、実行中に実行例外を生じず、プログラム制御順序でこれに先行するすべての発行済み命令がリタイアした発行済み命令をそれぞれリタイアする。予め定義した実行例外の最初の1つが発行済み命令によって生じた時、実行手段は発行済み命令の実行を継続し、精確状態手段は、リタイアできない発行済み命令に遭遇するまで発行済み命令のリタイアを継続することによって実行トラップの順序付けを行う。リタイアできない発行済み命令は、(a)

第1の実行例外を生じさせた発行済み命

令、および(b)第1の実行例外を生じさせた発行済み命令より早く発行されたが、第2の実行例外を第1の実行例外より遅く発生させた発行済み命令の1つである。浮動小数点状態レジスタは浮動小数点トラップタイプフィールドを有する。書き込み手段は、浮動小数点状態レジスタの浮動小数点トラップタイプフィールドにデータを書き込み、リタイアできない命令に対応する記憶要素の浮動小数点識別フィールドのデータが、リタイアできない命令が浮動小数点命令であることを示す時、リタイアできない命令に対応する記憶要素の浮動小数点トラップタイプフィールドのデータによって、浮動小数点実行トラップの種類を識別する。

前記問題は、本発明の構造および方法の別の側面によって解決されるが、これは、投機的にトラップを取り、トラップから戻るためのデータプロセッサと関連する方法を提供する。データプロセッサは、それぞれ対応するトラップレベルを有する入れ子トラップを取るため所定の数のトラップレベルをサポートする。データプロセッサは、チェックポイント形成手段、チェックポイントへのバックアップ手段、トラップを取る手段、トラップから戻る手段、レジスタおよびトラップスタックユニットからなる。レジスタは、トラップを取る度にデータプロセッサの状態を定義する内容を持つ。トラップスタックユニットは、トラップレベルより数の多いトラップスタック記憶記述項を有するトラップスタックデータ記憶構造を含む。また、トラップレベルの1つに現在マッピングできるトラップスタック記憶記述項の現在利用可能リストを保持するフリーリストユニットを含む。フリーリストユニットは、トラップを取る度に、対応するトラップレベルの1つに現在マッピング可能な次のトラップスタック記憶記述項を識別する。トラップスタックユニットはさらに、トラップを取る度に、現在利用可能なトラップスタック記憶記述

項の次の1つにレジスタの内容を書き込む読み取り／書き込みロジックを含む。さらにまた、トラップスタック記憶記述項の1つに各トラップレベルの現在のマッピングを保持するリネームマッピングロジックを含む。リネームマッピングロ

ジックは、トラップを取る度に、トラップスタック記憶記述項の1つへの対応するトラップレベルの古いマッピングを、現在利用可能なトラップスタック記憶記述項の次のものへの対応するトラップレベルの現在のマッピングに置き換える。トラップスタックユニットはまた、現在のマッピングによってトラップレベルの1つに現在マッピングされていないが、トラップレベルの1つにマッピングできないトラップスタック記憶記述項の利用不可リストを保持するリソースリクレームユニットも含む。リソースリクレームユニットは、トラップを取る度に、古いマッピングによって対応するトラップレベルにマッピングされたトラップスタック記憶記述項を利用不可リストに加え、取ったトラップを元通りにできなくなる度に、古いマッピングによって対応するトラップレベルにマッピングされたトラップスタック記憶記述項を利用不可リストから取り除く。フリーリストユニットは、利用不可リストから取り除かれたトラップスタック記憶記述項を、現在利用可能リストに加える。最後に、トラップスタックユニットは、チェックポイント記憶記述項を含むチェックポイント記憶ユニットを含む。形成された各チェックポイントは、対応するチェックポイント記憶記述項を有し、形成された各チェックポイントについて、チェックポイント記憶ユニットがリネームマッピングロジックの現在のマッピングとフリーリストユニットの現在利用可能リストを、対応するチェックポイント記憶記述項に記憶できるようにする。チェックポイントへのバックアップの度に、リネームマッピングロジックは、現在同ロジックが保持するマッピングを、対応するチェックポ

イント記憶記述項に記憶されたマッピングに置き換え、フリーリストユニットは、現在同ユニットが保持する利用可能リストを、対応するチェックポイント記憶記述項に記憶された利用可能リストに置き換える。

前記問題は、本発明の別の側面によって解決されるが、これは、複数の投機的発行・予測された命令の実行結果を同時にモニターするためのデータプロセッサおよび関連する方法を提供する。本発明の構造および方法は、プロセッサ内の1つ以上の実行ユニットからの実行結果信号を受信するため連結されたウォッチポイントデータを記憶するための複数のウォッチポイントレジスタを有するウォッ

チポイントユニットを提供する。投機的に発行・予測された命令の予測された条件データ結果を含むウォッチポイントデータを記憶するウォッチポイントレジスタは、命令が発行される時に割り当てられる。予測された条件データ結果は、投機的に発行された命令の制御フロー転送方向が決まる条件の予測値と、投機的に発行・予測された命令が投機的に発行されたベースを識別する。ウォッチポイントユニットは、実行ユニットからデータフォワードバスで転送される投機的に発行・予測された命令の実行結果信号をモニターし、記憶されたウォッチポイントデータと、実際の既知条件データ結果信号を初めとする実行結果信号および、実際の条件コードが予測された条件コードと一致するか否かを決定する所定の規則に基づき、所定の事象の発生を検出する。投機的に発行・予測された命令の1つについてウォッチポイントレジスタに記憶されたウォッチポイントデータを、投機的に発行された命令に関連するデータフォワードバスで到着した結果信号と比較し、信号が一致するか一致しないかを決定する。一致は、投機的に発行・予測された命令が正しく予測されたことを示し、不一致は、投機的に発行・予測された命令が誤予

測だったことを示す。誤予測が識別されると、プロセッサは初期状態に戻り、誤予測に基づいて実行された命令を元通りにする。本発明の側面は、複数の実行ユニットと、複数の予測された分岐またはジャンプ・アンド・リンク命令を初めとする複数の投機的に発行された命令を同時にモニターする能力を含む、実行ユニットからの条件コードデータを同時につかむ構造および方法；実行ユニットからのデータフォワードバスを同時にウォッチすることで、予測された分岐またはジャンプ・アンド・リンクが待っている複数の条件コードデータまたは計算されたジャンプアンド・リンクアドレスをグラブする構造および方法；分岐またはジャンプ・アンド・リンク命令の複数の誤予測信号を同時に生成する構造および方法；1つの共有記憶領域に別の分岐アドレスまたは予測されたジャンプ・アンド・リンクアドレスを記憶する構造および方法；および条件コードタグ比較とグラビングのタスクを分けることで、クリティカルパスをスピードアップするための構造および方法を提供する。本発明の構造および方法の1実施例では、これら複数

の側面を有利に組み合わせて全体的なプロセッサ性能を向上させる。

図面の簡単な説明

図 1 は、投機的プロセッサの精確状態を維持するために再順序付けバッファを用いる従来のアプローチを示す。

図 2 は、命令を実行する前にチェックポイントに状態を記憶し、後に記憶したチェックポイント情報から状態を復元することによって、投機的プロセッサに精確状態を維持する従来のアプローチを示す。

図 3 は、図解のために略図で命令の例示シーケンスと従来のチェックポイントの構造と内容を示すもので、特定のマシンの実際のチ

ェックポイントデータを表すものではない。

図 4 は、中央処理装置を含む本発明のデータプロセッサの実施例のトップレベル機能ブロック図である。

図 5 は、一部典型的命令タイプに関する C P U の新規命令パイプラインの実施例の段階である。

図 6 は、命令のプリフェッチとキャッシュユニットおよびフェッチユニットを含む本発明の C P U の例示分岐ブロック (B R B) を示す。

図 7 は、発行ユニット (I S U) 、精確状態ユニット (P S U) 、フリーリストユニット、制御レジスタファイルを含む例示発行ブロック (I S B) を示す。

図 8 は、発行ユニットがレジスタリネーミングフリーリストユニットから受取り、本発明のレジスタリネーミングの実施に用いる例示信号を示す。

図 9 は、精確状態ユニット (P S U) の実施例を示す。

図 1 0 は、発行 / コミット / リクレームユニット (I C R U) の実施例のコンポーネントと、その動作に関連する入出力信号の機能ブロック図である。

図 1 1 は、命令状態情報とデータ構造に記憶される第 1 の例示データセットに関連する各種ポインタを記憶するためのアクティブな命令データ構造とメモリ命令データ構造の実施例の略図である。

図 1 2 は、A - リングビットセット / クリアロジックの一部構造的コンポーネントを含む例示的状态制御ロジックユニットを示す。

図 1 3 は、状態を追跡し精確状態を維持するため命令タグを使用する本発明の方法の実施例のフローチャートである。

図 1 4 は、精確状態の維持に命令状態を追跡する本発明の方法の実施例の模式的フローチャートである。

図 1 5 は、精確状態の維持に命令状態を追跡する本発明の方法の実施例に従って、アクティブな命令リングとメモリ命令リングに状態情報を書き込む方法のフローチャート図である。

図 1 6 は、フィジカルレジスタを含むフィジカルレジスタファイルの入ったレジスタファイルおよびリネームユニット (R F R N) の実施例を示す。

図 1 7 は、レジスタリネーミングと関連する制御レジスタファイルの実施例を示す。

図 1 8 は、リソースリクレームファイル (R R F) の実施例を示す。

図 1 9 は、本発明のレジスタリネーミング方法を示す。

図 2 0 は、命令発行、非起動、コミットおよびリタイアを含む精確状態の維持のための本発明の方法の実施例のフローチャートである。

図 2 1 は、命令状態情報とデータ構造に記憶された第 2 の例示データセットに関連する各種ポインタを記憶するためのアクティブ命令データ構造およびメモリ命令データ構造の実施例の略図である。

図 2 2 は、C P U のデータフォワードブック内のロード／ストアユニット (L S U) の機能ブロック図である。

図 2 3 は、精確状態を維持しながら、ロード／ストア命令を含む長待ち時間命令を積極的にスケジューリングするための本発明の方法の実施例のフローチャートである。

図 2 4 は、チェックポイントユニットの機能ブロック図である。

図 2 5 は、ウォッチポイントユニットの機能ブロック図である。

図 2 6 は、タイムアウトチェックポイントを形成するためのオプションのチェックポインティング強化を含む本発明のチェックポインティング方法の実施例のフローチャートである。

図27は、ウォッチポイントユニットの実施例内の主要機能ブロックの機能ブロック図である。

図28は、条件コードグラビングロジック、評価ロジック、ウォッチポイント要素記憶および制御ユニット、およびターゲットRAMの特定の実施例を示すウォッチポイントユニットの実施例である。

図29は、カレントマッチングロジック、アレーマッチングロジック、および評価レディおよび評価条件コードロジックを含む条件コードセレクトロジックの入ったウォッチポイントの実施例の一部を示す。

図30は、分岐予測および評価に関連する例示タイミングチャートを示す。

図31は、カレントマッチロジックおよびアレーマッチロジックの実施例に関連する構造的詳細を示す。

図32は、条件コードセレクトロジックの実施例に関連する構造的詳細を示す。

図33は、評価レディロジックの実施例に関連する構造的詳細を示す。

図34は、評価ツルーロジックの実施例に関連する構造的詳細を示す。

図35は、ターゲットRAMおよびジャンプ・リンク命令評価ロジックの実施例に関連する構造的詳細を示す。

図36は、複数同時未解決分岐評価のための本発明のウォッチポイント方法の実施例のフローチャートである。

図37は、浮動小数点例外（FPXCEP）データ構造の実施例の略図である。

図38は、例示浮動小数点例外リングデータ構造の信号インターフェースを示す。

図39は、RDおよびAEXCロジックを含む浮動小数点例外ユニットを示す。

図40は、バックアップおよびバックステップコンポーネントを含むバックトラックユニットの実施例の図である。

図41は、バックアップおよびバックステップ手順を含む命令境界で精確状態

を維持および回復するための本発明の方法の実施例のフローチャートである。

図42は、レジスタリネームファイルにロジカルからフィジカルマッピングを復旧する方法を示す。

図43は、チェックポイント境界への1つのマシンバックアップの後の命令境界への2つのマシンバックステップを含むマシンバックトラックの例である。

図44は、例示プライオリティロジックおよび状態マシン (P L S M) を示す。

図45は、例示トラップスタックユニットを示す。

図46は、トラップスタックの記憶要素リストを記憶するためのフリーリストロジックの実施例を示す。

図47は、例示リネームマップロジックを示す。

図48は、例示トラップスタック R R F を示す。

図49は、トラップスタックチェックポイント記憶ユニットを示す。

図50は、ウォッチポイント W P _ _ A C T I V E _ _ V E C および W P _ _ M I S P R E D ロジックを示す。

図51は、トラップスタックバックアップ例を示す。

図52は、X I C C 書き込みロジックを示す。

図53は、X I C C グラビングロジックおよびアレーレーターマ

ッチロジックを示す。

図54は、ウェイトー・フォー条件コード待ちロジックを示す。

実施例の説明

図4は、データプロセッサ50のトップレベル機能ブロック図である。データプロセッサは、新規の順序無視投機的実行 (スーパスカラ) 中央処理装置 (C P U) 51を含む。例示 C P U は、SPARC-V9 Architecture Manualに記載されたSPARC-V9命令セットを実行可能である。D.WeaverおよびT.Germondによる「The SPARC Architecture Manual」、Version 9、Englewood Cliffs (1 9 9 4 年) は、参照により本書に明示的に組み込む。但し、当業者は、本書に記載する新規設計および方法がSPARC-V9アーキテクチャに限定されないことを認識するであろう。

実際、CPU 51は、すべてのデータプロセッサに該当する精確マシン状態を維持しながら、スループットを上げる新規設計および方法を採用している。特に、これは(1)プロセッサリソースおよび状態をモニターし、回復する命令をトラッキングするための新規設計及び方法；(2)ロード／ストア命令などの長待ち時間命令を積極的にスケジューリングするための新規設計および方法；(3)マシン状態をチェックポイントニングするための新規設計および方法；(4)例外または誤予測検出後にマシン状態を回復するための新規設計および方法を採用している。

本明細書の実施例の説明セクションの概略を便宜上ここに示す。本明細書の見出しは便宜的参照のためにのみ示すもので、本セクションの開示の適用可能性を本発明の特定の側面または実施例に限定すると解釈してはならない。

I. CPUオペレーション概要

II. 命令トラッキング

A. 命令フェッチ

B. 命令発行

1. 命令発行概要

2. 命令シリアル番号割り当てと命令状態記憶

a. 命令発行ーコミットーリクレームユニット(ICRU)

b. 命令発行関連ポインタおよび発行段階中のポインタ維持

3. レジスタリネーミング

4. リザベーションステーションへのディスパッチ

C. 命令実行および完了

D. 命令実行段階へのPSU参加

E. 命令コミットメント

F. 命令リタイアおよびリソース回復

1. 命令コミットおよびリタイア時のRRFへの更新とマップリネーム

2. 命令リタイア時のRRF読み取り

3. ポインタを使った精細状態維持

III. 積極的長待ち時間（ロード／ストア）命令スケジューリング

A. メモリ命令状態情報記憶

B. 長待ち時間情報のトラッキングおよびスケジューリングのためのNMCSN
およびPBSNポインタ

C. NMCSN前進

D. データフローブロック内のロードストアユニット（LSU）

1. インレンジメモリ参照（長待ち時間）命令の識別

2. 積極的長待ち時間（ロード／ストア）命令スケジューリングのための基本
的構造および方法強化

IV. チェックポインティング

A. チェックポイント割り当てレジスタの構造

B. チェックポイント割り当て

C. チェックポイントリタイア

D. マシンバックアップでのチェックポイント維持

E. タイムアウトチェックポイント強化

F. あらゆる命令境界での精細状態維持と回復

G. チェックポイントされたデータ量を減らすため所定の命令についてマシンを
同期させる方法H. チェックポイントされたデータ量を減らすためレジスタリネームマップをチ
ェックポインティングする方法

V. 誤予測および例外からの回復

A. 同時複数未解決分岐／ジャンプ・アンド・リンク命令評価のためのウォッチ
ポイントによる誤予測検出

1. ウォッチポイント要素の起動

2. データフォワードバスから直接CCデータのウォッチポイントグラビング

3. ウォッチポイントオペレーションの例

4. 分岐命令の評価

5. ジャンプーリンク（JMP L）命令の評価

B. 例外検出

1. 発行トラップ検出
2. 実行トラップ検出

C. プロセッサを初期状態にバックトラックすることによる回復

1. チェックポイント境界へのプロセッサバックアップ
2. 誤予測された命令、REDモード、および実行トラップ開始バックアップ
3. あらゆる命令境界へのプロセッサバックステップ

D. 例外&誤予測回復中のプライオリティロジックおよび状態マシンオペレーション

E. トラップスタックによるトラップの取扱

I. CPUオペレーション概要

CPU 51は、レベル1 (L1) 命令キャッシュ52から命令をフェッチし、レベル1 (L1) データキャッシュ53にデータを記憶し、ここからデータを取り出す。L1命令およびデータキャッシュは、CPU 51と同じチップ/基板に物理的に配置することもできるが、異なるチップ/基板に組み立てることもできる。

CPUはまた、メモリ管理制御ユニット(MMU) 54とインタラクトして、データプロセッサ50の全体状態を決定する。特に、MMUは外部メモリ56から命令を取り出し、外部メモリ56にデータを記憶し、ここからデータを取り出し、外部I/Oデバイス57からデータを受取り、ここへデータを出力し、外部診断プロセッサ58、CPU 51、キャッシュ52および53に診断情報を提供し、ここから診断情報を受取り、データアドレス翻訳を実行し、メモリ状態情報を記憶・追跡する。

図5は、予測されたプログラム制御命令、固定および浮動小数点命令、およびロード/ストア命令を含む一部の典型的な命令タイプに関する、CPU 51の新規命令パイプラインの段階（および関連するマシンサイクル）を示す。このパイプラインを実行するため、図4に示すように、CPU 51は分岐ブロック(BRB) 59、発行ブロック(ISB) 61およびデータフローブロック(DFB)

6 2 を含む。

図 4 および図 5 を参照すると、B R B 5 9 はフェッチ段階中に命令をフェッチし、これらの命令を I S B 6 1 に与える。B R B 5 9 は、プログラム制御命令の各種タイプについて、ターゲットアドレ

ス予測と分岐取得・不取得予測を行う。そのため、これはフェッチ段階中に投機的に命令をフェッチすることができる。

I S B 6 1 は、命令が投機的にフェッチされた可能性があるため、発行段階中に予測されたプログラムカウンタ（P C）順でフェッチされた命令を発行する。これはまた、予め定義された間隔に当たる命令と、B R B 5 9 が予測を行ったプログラム制御命令を含む予め定義された種類の発行された命令について、発行段階中に C P U 5 1 のマシン状態のチェックポイントを形成する。

B R B 5 9 は、I S B 6 1 による発行と同じ段階で、n o p と予測されたプログラム制御命令を実行、完了する。但し、D F B 6 2 は、必要なリソースが利用可能になり次第、発行された固定小数点および浮動小数点命令を実行、完了する。さらに、D F B 6 2 は、必要なリソースが利用可能になった時、実行・完了した時点で例外（例：発行トラップ、実行トラップまたは割り込み）やプログラム制御誤予測の場合に元通りにする必要のないやり方で、発行されたロード／ストア命令を実行のために積極的にスケジューリングする。そのため、一部の固定小数点、浮動小数点、およびロード／ストア命令の発行段階が他のものより早くても、必要なリソースはまだ利用可能になっていないため、実行および完了段階が後になる。さらに、これらは B R B 5 9 による初期のプログラム制御予測に基づき、B R B 5 9 が投機的にフェッチした可能性があるため、投機的に実行および完了されたかもしれない。言い換えると、D F B 6 2 に発行された命令は、予測された P C 順序を無視して実行・完了されることがある。

命令がエラー発生（例：発行トラップ、実行トラップまたは誤予測）なしに完了すると、非起動段階で I S B 6 1 によって非起動にされる。命令は予測された P C 順序を無視して完了することがある

ので、誤予測の P C 順序を無視して非起動とされることがある。

そこで、非起動命令は、コミット段階に実際の P C 順で I S B 6 1 によってコミットされる。非起動の命令は、先行する発行済み命令をすべてコミットしていなければコミットされないので、真である。その結果、命令がコミットされると、例外または誤予測の場合、元通りにできない。これはすなわち、この点までの実際の P C 順序が正しく、命令は実際の P C 順でコミットされたことを意味する。さらに、チェックポイントを形成した命令をコミットすると、先行のチェックポイントをリタイアすることができる。

命令をコミットすると、リタイア段階に実際の P C 順で I S B 6 1 によってリタイアされる。命令リタイア後、それに割り当てられたリソースはリクレームされ、他の命令が発行された時にそれらに再割り当てすることができる。

I S B 6 1 は、パイプライン中のあるポイントで発生する実行またはプログラム制御の誤予測からの回復を行う。そのため、プログラム制御誤予測が発生すると、I S B 6 1 は、不正プログラム制御命令が発行された時、形成したチェックポイントに C P U 5 1 をバックアップする。同様に、実行トラップが発生すると、I S B 6 1 はまず、不正命令発行後にチェックポイントが形成されていれば、最も早いチェックポイントに C P U 5 1 をバックアップし、および／または不正命令に C P U 5 1 をバックステップする。チェックポイントへのバックアップおよび／またはバックステップにおいて、C P U 5 1 は、不正命令の発行および実行直前に存在した正しいマシン状態に戻る。

I S B 6 1 が、プログラム制御誤予測を生じさせた不正命令にバックアップされていると、B R B 5 9 は、正しいプログラムカウンタ値と正しいマシン状態で命令のフェッチを始める。しかし、I S

B 6 1 が実行トラップを生じた命令にバックアップされているか、および／またはバックステップされていると、B R B 5 9 に、適切なトラップ取扱ルーチンのターゲットプログラムカウンタ値が与えられる。そこで、トラップ取扱ルーチンはトラップを処理し、不正命令のプログラムカウンタ値を戻すか、次の命令をフェッチするため B R B 5 9 に次のプログラムカウンタ値を戻す。不正命令のプロ

グラムカウンタは、トラップ取扱ルーチンがCPU 51に不正命令をフェッチし、その発行と実行を再試行するよう命じると、戻される。しかし、トラップ取扱ルーチンがCPU 51に不正命令後に次の命令をフェッチして、不正命令の発行と実行をスキップするよう命じると、次の不正命令のプログラムカウンタが戻される。

II. 命令トラッキング

前に言及したように、CPU 51は、CPU 51の命令パイプラインに命令トラッキングのための新規設計および方法を採用している。パイプラインは、この設計および方法を実施するため、図5に示し、前に簡単に述べたように、フェッチ、発行、実行、完了、非起動、コミットおよびリタイアの段階を含む。

A. 命令フェッチ

再び図4を参照すると、BRB 59はフェッチ段階中に命令をフェッチし、これらをISB 61に与える。図6に示すように、BRB 59は、命令プリフェッチおよびキャッシュユニット(IPCU) 100とフェッチユニット102を含む。

IPCU 100は、レベル1(L1)命令キャッシュ52から1度に4個の命令(INSTs)を取り出すことができる。取り出された命令は、IPCU 100の命令レコーダによって、BRB 59、ISB 61およびDBF 62の利用により適したフォーマットにレコードされる。別の実施例では、IPCU 100は、1度に4個

より多いか少ない命令を取り出して実施することもできる。

フェッチユニット102は、フェッチプログラムカウンタ(FPC)値、アーキテクチャルプログラムカウンタ(APC)値、および次のアーキテクチャルプログラムカウンタ(NAPC)値を計算するプログラムカウンタ(PC)ロジック106を含む。各マシンサイクルで計算されたFPC、APCおよびNAPCの値はそれぞれ、PCロジックのFPC、APCおよびNAPCレジスタ112から114に記憶される。FPCレジスタの現在のFPC値は、現在のマシンサイクルでフェッチされている命令を指すが、APCおよびNAPCレジスタ

113および114の現在のAPCおよびNAPC値は、現在のマシンサイクルでISBが発行可能な最初の命令と、前回のマシンサイクルでフェッチされた次の命令を指す。

FPC値に対応して、1度に4個の命令(F__INSTs)がIPCU100によってフェッチされる。別の実施例では、1度に4個より多いか少ない命令をフェッチするよう実施することができる。

各FPC値について、分岐履歴表104には、FPC値でフェッチした各命令の分岐予測フィールド(BRP)が含まれる。そのため、BRPフィールドはそれぞれFPC値でフェッチした命令の内の1つに対応する。フェッチした命令のいずれかが、SPARC-V9B Pcc、Bicc、BPr、FBfccおよびFBPfcc命令など条件付き分岐命令である場合、これらの対応するBRPフィールドは、分岐を、分岐命令のこの後の発行、実行および完了段階で予測すべきか否かを識別する。分岐履歴表104は、FPC値に対応してフェッチした命令にBRPフィールドを出力する。そこでBRPフィールドは、フェッチユニット102によってフェッチした対応

する命令(F__INSTs)に追加され、フェッチ回転レジスタ110が受け取る合成命令を与える。

フェッチレジスタには、1度に4個の命令を保持するため4個のラッチがある。このラッチは、予測されたPC順で、ISU200が発行できる次の4個の命令の入った4個のスロット(0-3)を有する発行ウィンドウを定義する。但し、前回のマシン中にフェッチレジスタの保持した命令の一部は、これから説明するような発行上の制約のため、ISU200が発行していないことがある。フェッチされた命令が予測されたPC順で発行されるようにするため、ISU200はフェッチレジスタ制御(FCH__REG__CTRL)信号を生成するが、この信号は、最後のマシンサイクルで発行されなかった命令を該当するラッチに入れるようフェッチレジスタを制御することで、発行ウィンドウ内で予測されたPC順にし、ISU200が予測されたPC順で発行できるようにする。さらに、FCH__REG__CTRL信号に対応して、フェッチレジスタは残りのラッチ

に、予測されたPC順で次に来るフェッチしたばかりの命令を入れる。

別の実施例では、フェッチレジスタは、1度に4個より多いか少ない命令を記憶できるよう実施できる。さらに、例示CPU51ではフェッチユニット102に1個のフェッチレジスタとして説明したが、今説明した種類のフェッチレジスタは、デコードを実行するCPUの各ブロックに使うことができる。そして、フェッチレジスタをこれらブロックに設置して、デコードした命令をフェッチレジスタに記憶する前、フェッチサイクル中に初期デコードを行い、残りのデコードを発行段階に行うようにすることができる。

B. 命令発行

1. 命令発行概観

ISB61は、各発行段階中にフェッチした命令を発行する。図7に示すように、ISB61は、発行ユニット(ISU)200、精細状態ユニット(PSU)300、フリーリストユニット700、および制御レジスタファイル800を含む。各発行段階中、ISU200は、BRB59のフェッチレジスタ110から、一度に4個の命令(F__INSTs__BRP)を受け取る。そして、これらをデコードして、その内いくつを発行するかを、次に説明するような各種発行上の制約に基づいて決定する。

フェッチレジスタ110は、各発行段階中、ISU200に一度に4個の命令を与えるため、ISU200は各発行段階中、最大4個の命令を発行することができる。しかし、各発行段階の発行ウィンドウで予測されたPC順で4個の命令を発行することしかできない。そのため、他の発行上の制約から、現在の発行ウィンドウで命令の内の1つが発行できない場合、この命令の前の発行ウィンドウスロットの命令しか、現在の発行段階では発行できない。別の実施例では、フェッチレジスタ110は、1つの発行段階あたり4個より多いか少ない命令をISU200に与えるよう構築されているため、ISU200は、1つの発行段階あたり4個より多いか少ない命令を発行するよう構築することができる。図8を参照すると、FPU600、FXU601、FXAGU602およびLSU603は、発行された命令について、これらにディスパッチされた命令データを記憶す

るためのリザーベーションステーションあるいは待ち行列を有する。但し、発行段階中、リザーベーションステーションの一部は、ディスパッチされた命令データを記憶できるだけの記憶要素または記述項を持たないことがある。そのため、FPU600、FXU601、FXAGU602およびLSU603は、それぞれFPU600、FXU601、FXAGU602およびLSU603

のリザーベーションステーションに、ディスパッチされた命令データを受け取ることのできる記述項がいくつあるかを示すENTRIES__AVAILABLE信号を出力する。その結果、ISU200は、ENTRIES__AVAILABLE信号で示された利用可能な記述項の数に基づき、浮動小数点、固定小数点、およびロード・ストア命令を発行する。

さらに、例示CPU51では、FXU601が、固定小数点計算に関わるプログラム制御、乗算／除算、および制御レジスタ読み取り／書き込み命令を実行するDFB62の唯一の実行ユニットである。そのため、この場合、ISU200はFXU601による実行のためこれらの種類の命令を発行する。別の実施例では、FXAGU602も、固定小数点データに関わるプログラム制御、乗算／除算、および制御レジスタ読み取り／書き込み命令を実行するよう構成することができる。この場合、ISU200も、FXAGU602による実行のためこれらの種類の命令を発行することになる。さらに、実行ユニット600から603はそれぞれ1以上の機能ユニットからなり、各機能ユニットは発行された命令を実行することができる。そのため、発行することのできる命令の数は、各実行ユニット内の機能ユニットの数で決まる。

図7を参照すると、PSU300は、ISU200による命令発行に発行上の制約を課すこともできる。PSU300はISSUE__KILL信号を出力することができ、これは、フェッチレジスタ110から受け取った命令の内どれを現在の発行段階中に発行してはならないかを識別するものである。後により詳しく説明するように、例外検出後か、次に説明する同期命令の実行、完了、非起動およびリタイア中に、PSU300が不正命令にバックアップまたはバックステップしている時、ISSUE__KILL信号がアサート

される。

特定の種類の命令については、I S U 2 0 0 は、これら同期命令の内1つを発行する前に、C P U 5 1 が同期されることを確かめる。C P U 5 1 は、特定の命令に先行するすべての命令が発行、実行、完了、非起動およびリタイアした時、その命令について同期する。そのため、I S U 2 0 0 がフェッチレジスタ110から受け取った命令の内1つが同期命令であると決定すると、その前の発行ウィンドウスロットにある命令を発行し、C P U 5 1 が同期したことを示すM A C H I N E _ S Y N C 信号をP S U 3 0 0 から受け取るまで待つ。これが発生すると、命令は最も早いP C 値（すなわちスロット0）で命令の発行ウィンドウのスロットに入り、I S U 2 0 0 はただそれを発行する。例示C P U 5 1 は、特定の命令種類について、前に説明したマシン状態同期方法を実施するが、別の技術では、コミットした命令は非投機的であることが保証されるため、前回の命令のコミット後に同期命令の発行ができる。

I S U 2 0 0 はまた、SPARC-V9 Architecture Manualに説明するような、命令の発行に影響する発行トラップを検出する。I S U 2 0 0 は、制御レジスタファイル800からレジスタ（C N T R L R E G）フィールドを受取り、フェッチレジスタ110から受け取った命令（F _ I N S T s _ B R P s）をデコードして、発行段階中に一定の種類の発行トラップが発生したか否かを検出する。SPARC-V9アーキテクチャに基づく実施例では、これはSPARC-V9 Architecture Manualに従って行われる。さらに、他の種類の発行トラップはレジスタ制御を必要とせず、I S U 2 0 0 のデコードした命令操作コードに基づき、取得・検出される。

最も早い発行ウィンドウスロットの命令によって生じた発行トラップのみが取得される。そのため、1以上の発行トラップが検出さ

れると、最早スロットの命令の原因となった発行トラップのスロットより前の発行ウィンドウスロットの命令のみ、I S U 2 0 0 によって発行することができる。さらに、しばしば発生する発行トラップについては、C P U 5 1 は、前に述べたような方法で同期し、発行トラップを投機的に取得しないようにする。別の実施例では、発行トラップがスロット0でのみ発生するようC P U 5 1 を構成して

、ロジックを減らし、簡素化することができる。

各発行段階中、I S U 2 0 0は発行される各命令にシリアル番号を割り当て、これらシリアル番号(S N s)をD F B 6 2にディスパッチする。後でより詳細に説明するように、P S U 3 0 0は、割り当てられたシリアル番号を使って発行された命令を記録する。さらに、例示C P U 5 1では、発行されたがまだリタイアしていない命令を一度に所定の数だけ記録することができる。しかし、発行された命令をリタイアすると、それらのシリアル番号が利用可能となる。そのため、P S U 3 0 0は、現在の発行段階にいくつのシリアル番号が利用できるかを示す信号を含むS N _ A V A I L信号と、これら利用可能なシリアル番号を含む信号を、I S U 2 0 0に与える。現在の発行段階中、S N _ A V A I L信号が、現在利用可能なシリアル番号の数が発行できる命令の数より少ないことを示すと、シリアル番号を割り当てることのできる最も早いスロットの命令だけが実際に発行される。

やはり発行段階中、I S U 2 0 0は、ある命令についてC P U 5 1のマシン状態のチェックポイントを形成すべきか否かを決定する。後でより詳細に説明するように、チェックポイントは一定の種類の命令について、所定の発行段階間隔で形成される。そのため、発行する命令にチェックポイントが必要とI S U 2 0 0が決定し、および／またはP S U 3 0 0が、チェックポイントを形成した最後の

発行段階から所定数の発行段階が発生したことをT I M E O U T C H K P T信号によって示すと、I S U 2 0 0はD O _ C H K P T信号を生成し、P S U 3 0 0、制御レジスタファイル8 0 0、B R B 5 9およびD F B 6 2に、M C P U 5 1のマシン状態のチェックポイントを形成するよう指示する。

さらに、I S U 2 0 0は、発行した各命令にチェックポイント番号を割り当て、これらチェックポイント番号(C H K P T _ N s)をD F B 6 2にディスパッチする。そして、チェックポイントを形成した各命令に、新しいチェックポイント番号が割り当てられる一方、チェックポイントを形成しない命令には、前のチェックポイントのチェックポイント番号が割り当てられる。P S U 3 0 0は、割り当てられたチェックポイント番号を使って形成されたチェックポイントを記録

する。

割り当てられたシリアル番号同様、P S U 3 0 0 は、形成されたがリタイアしていないチェックポイントを一度に所定の数だけ記録する。但し、チェックポイントがリタイアされると、チェックポイント番号が利用可能になるため、P S U 3 0 0 は、現在の発行段階にいくつかのチェックポイント番号が利用できるかを示す信号を含む C H K P T _ A V A I L 信号と、利用可能なチェックポイント番号を含む信号を、I S U 2 0 0 に与える。そのため、C H K P T _ A V A I L 信号が、現在利用可能な新しいチェックポイントの数が新しいチェックポイントを形成しなければならない命令数より少ないことを示すと、新しいチェックポイント番号を割り当てることのできる最も早い発行ウィンドウスロットの命令だけが実際に発行される。

さらに、C P U 5 1 は、1 つの発行段階につき所定の数 of チェックポイントを形成するようにのみ構成することができる。そのため

、たとえば C P U 5 1 が 1 つの発行段階につき 1 つしかチェックポイントを形成できず、チェックポイントを形成する必要のある発行ウィンドウに 2 個の命令がある場合、I S U 2 0 0 は、現在の発行段階中、最も早いスロットにある命令のみ発行し、別の発行段階の他の命令を発行する。

後で説明するように、C P U 5 1 は、プロセッサのスループットを上げるためレジスタリネーミングを採用している。レジスタリネーミングを正しく実施するため、I S U 2 0 0 はレジスタリネーミングフリーリストユニット (F R E E L I S T) から R E G S _ A V A I L 信号を受け取る。図 8 を参照すると、これら信号には、固定小数点レジスタファイルおよびリネームユニット (F X R F R N) 6 0 4 のいくつかの物理的固定小数点レジスタをリネーミングに利用できるか、浮動小数点レジスタファイルおよびリネームユニット (F P R F R N) 6 0 5 のいくつかの浮動小数点レジスタをリネーミングに利用できるか、浮動小数点状態および条件コードレジスタファイルおよびリネームユニット (F S R / C C R F R N) 6 0 6 のいくつかの物理的整数および浮動小数点条件コードレジスタをリネーミングに利用できるかを示す信号が含まれる。図 7 を参照すると、フェッチレジ

スタ110から受け取った命令のいずれかがレジスタリネーミングを必要とし、物理的レジスタがリネーミングに必要であるが、REGS_AVAL信号によって利用できないことが示されると、ISU200はこれらの命令を発行できない。

前記発行上の制約は、発行決定の概念を例証するため、例示CPU51に関連して説明した。さらに、これまで説明した発行上の制約とは異なるが、本書に説明した発行決定の基本的概念から逸脱しない実施も存在できる。そのため、当業者は、発行上の制約が実施依存であり、本書に説明する発明は、これまで説明した発行上の制

約に限定されないことを認識するだろう。

2. 命令シリアル番号割り当てと命令状態記憶

例示CPU51は、命令が発行された時点から、命令の実行が完了して究極的にリタイアするまで、命令に関連する独自の識別タグを割り当てる。逐次シリアル番号は、命令識別タグとして便宜上用いられる。シリアル番号は各段階で用いられ、例外または誤予測が発生すると、例外フリー処理中およびCPU51回復中に実質的にすべてのブロックが用いる。命令状態情報はCPU51に記憶され、実行完了信号、実行エラー信号、および分岐命令誤予測信号等の状態の変化に対応して連続的に更新される。ISB61内の精細状態ユニット(PSU)300が、命令シリアル番号タグの割り当てと、他のCPU51ユニット、特にDFB62から受け取った信号に対応して命令状態のトラッキングを行う。

図9を参照すると、精細状態ユニット(PSU)300は機能的にISB61内にあり、(1)CPU51での命令の発行、実行、完了およびリタイア状態のトラッキング；(2)分岐誤予測回復の検出と開始；(3)命令シリアル番号、チェックポイント、およびウォッチポイントを含む一定のマシンリソースの利用可能性のトラッキング；および(4)一般例外取扱および制御、を行う。PSU300内のいくつかのユニットは、図9に示し、後により詳細に説明するように、この機能性を達成するため実施される。

PSU300は、発行されたすべての命令について、これらがリタイアされる

までアクティビティ状態情報を維持する発行／コミット／リタイアユニット（ICRU）301を含む。ICRU301は、ISU200が次の命令発行サイクルで用いる4個までの命令シリアル番号（SNs）を識別する信号（SN__AVAILABLE）をISU200に与える。ISUが命令を発行すると、ISU200は

、前のサイクルでISUに与えられた（SN__AVAILABLE）最大4個のシリアル番号のうちどれがISUによって有効に発行されたかを、前に示したように、どのシリアル番号が割り当てられ、どのシリアル番号が各命令に関連するかを示すISSUE__VALID信号の形でICRUに知らせる。発行上の制約によって、ISUが命令ウィンドウのすべての命令を発行する能力が制約を受けることを思い出してほしい。ICRU301は状態情報と状態情報へのアドレスポインタを記憶・維持し、命令の種類にかかわらず、ISU200が有効に発行した各命令の命令発行、実行、完了、非起動、コミット、リタイアフェーズ状態をトラックする。ISU200はまた、発行された命令の内どれがnopおよび一定のプログラム制御命令に関連し、発行と同じ段階で非起動が可能かを、ISSUE__NOP信号でICRU301に知らせる。ISSUE__NOPは、ポーズや遅延を導入するような、命令ストリームに挿入できる特定の「nop」命令に限定されない。CPU51外のロード／ストアその他命令参照メモリなどの長待ち時間命令を積極的にスケジューリングする際、ISU200は、スロット0-3の命令のどれが長待ち時間かを識別するISSUE__MEM信号の形で、ICRU301にも知らせる。（長待ち時間命令の積極的スケジューリングは、基本的な本発明の構造および方法の強化として別に説明する。）

a. 発行／コミット／リクレームユニット（ICRU）

発行／コミット／リクレームユニット（ICRU）はPSU300内で機能し、n-ビット逐次シリアル番号などの命令タグを割り当て、CPU51内部のメモリで命令状態情報のデータ記憶領域を定義・維持することにより、発行されたすべての命令のアクティビティ状態情報を維持する。図10に、ICRU301のコンポーネントおよびICRUオペレーションに関連する入出力信号の機能ブ

ロック図を示す。ICRU301は機能的に4つの領域で構成される。命令状態情報データ構造308は、命令状態情報の記憶と、データ構造制御ロジック309の信号に対応して状態を更新を行い、データ構造制御ロジック309は、他のCPU51ユニット、特にDFB62からの信号に対応する。ポインタデータ構造310は、発行、完了、リタイア等、CPU51の各種命令段階マイルストーンへの状態ポインタとして働く複数のシリアル番号を個別に記憶するデータ記憶領域を含む。これらシリアル番号ポインタは、後でより詳細に説明する。ポインタ制御ロジック311は、他のCPU51ユニットから受け取ったデータと共に、データ構造308に記憶された状態情報を評価し、シリアル番号ポインタを更新して、カレントCPU51状態を反映させる。ポインタ値は、ICRU301およびPSU300からCPU51の他ユニットへのグローバル出力として与えられる。

命令状態情報データ構造308は、アクティブ命令リングレジスタ（Aーリング）312からなり、オプションでメモリ命令リングレジスタ（Mーリング）324からなることもできる。メモリ命令リングレジスタは、積極的ロード／ストア命令スケジューリングに関連して本明細書中の他の部分で説明する。データ構造制御ロジック309は、Aーリングセット／クリアロジック313からなり、オプションでMーリングセット／クリアロジック325からなることもできる。ポインタ制御ロジック311は、ISN／NISN初期化および前進ロジック321からなり、次の命令シリアル番号割り当てロジック322と、CSN／RRP前進ロジック323は、オプションでNMC SN前進ロジック326、マシン同期生成ロジック327、バックトラックモードポインタ調整ロジック328からなる。オプション要素は本発明の基本的な構造および方法の強化

に関連し、後でより詳細に説明する。

発行／コミット／リクレームユニットの命令発行段階のオペレーションを、図10を参照して説明する。発行された命令に割り当てるためISU200から送られた一覧番号（SN__AVAILABLE）は、ポインタデータ構造310内のポインタ値に基づきICRUのシリアル番号割り当てロジックで選ばれる。CPU51

を初期化した時（パワーアップ時など）、ポインタデータ構造310内のポインタも初期化され、ICRUはまず、発行ウィンドウの命令について最初のシリアル番号をISU200に割り当てる（SNの0-3等）。

ISU200はデータ構造制御ロジック309にISSUE_VALID信号を送り、前回のサイクルで与えられたシリアル番号の内どれが（4個のSNの内どれが）ISU200によって有効に発行され、どの命令スロットのものをICRUに知らせる。セット／クリアAーリング制御ロジック313はこの情報とポインタレジスタ310内のポインタの値を使って、Aーリング312に命令状態情報を書き込む。CPU51の1実施例では、ISSUE_VALIDは4ビットベクトルで、ISU200が命令シリアル番号（SN_AVAIL）のどれを実際に使って命令を実際に発行した命令スロットを識別したかをICRUに知らせる。そこでICRUは、SN_AVAILとISSUE_VALIDに基づきどのSNを使ったかを決定できる。

アクティブ命令リング312の実施例を、図11の略図を参照して説明する。図11のデータ構造は例示なもので、本特許の内容から、このアクティブ命令状態レジスタの実施には様々なデータ構造を利用できることを当業者は理解するであろう。図11は、64ビットデータ構造として実施したアクティブ命令リング（Aーリング

）312を示す。（メモリ命令リング（Mーリング）324も略図として示し、長待ち時間命令の積極的スケジューリングの強化に関連して本明細書の他の部分で説明する。）Aーリング312内の64の各アドレス指定可能ロケーションは、CPU51の1つの命令シリアル番号に対応する。各アクティブビット（Aービット）のセット（「1」）またはクリア（「0」）状態は、命令がアクティブであるか（「1」）、インアクティブであるか（「0」）を示す。基本的に、発行されて、エラーなしに実行を完了し誤予測されずに非起動とさえると、命令は始動する。1つのマシンサイクルで発行され効果的に実行完了した一部の命令では、Aービットが命令発行でクリアされる。Aーリング312のアドレス指定可能ロケーションの数は、CPU内の同時未解決の命令の数を限定する。

例示プロセッサでは、Aーリング312は円形、環状あるいは循環データ構造として実施され、ここでポインタは「モジュロAーリングレジスタ長さ」タイプ演算（ここでは、モジュロ64タイプ演算）を使って、データ構造に沿って移動する。すなわち、ポインタが第1のデータ記憶場所（アドレス指定可能ビット0）から一連の中間記憶場所（ロケーション15、16...等）を通して最後のデータ記憶場所（アドレス指定可能ビット63）までインクリメンタルに移動すると、最初のロケーション（アドレス指定可能ビット0）に戻る。当業者は、本発明の内容から、環状データ構造は有利ではあるが、本発明にとって不可欠ではなく、本発明の特徴を実施するのに他のデータ構造も使えることを理解するであろう。さらに、例示Aーリングは必要な状態情報の維持に64個のシングルビットのアドレス指定可能ロケーションを持つが、これらおよび追加状態インジケータの記憶に、マルチビットアドレス指定可能ロケーションを設けることもできる。たとえば、ある実施例では、任意回転

の64ビットレジスタを用い、第2の実施例では、8個の8ビットレジスタを用いて64個のAーリング312Aービットとしている。単純なデコーディング回路が、後に説明するようにビット書き込み（セットとクリア）を可能にする。

ISU200による「命令発行」の結果、命令に命令シリアル番号が割り当てられる。割り当て可能なシリアル番号の数は、Aーリング312のアドレス指定可能ロケーション（ビット位置等）の数によって限定される。そのため、各アドレス指定可能ロケーションは独自のシリアル番号に対応する。たとえば、例示64ビット環状Aーリングには、64個までの命令シリアル番号を同時に割り当てることができる。Aーリングのロケーションを多くか少なくすることによって、これより多くか少ないシリアル番号を割り当てることができる。すべてのシリアル番号を割り当てると、シリアル番号とAーリングのロケーションがその後のマシンサイクルで解放されるまで、それ以降の命令発行をISU200によって機能停止しなければならない。ISU200は、SN-AVAIL信号によってSNが利用可能か否かを知らされる。Aーリング312の各アドレス指定可能ロケーションは利用可能なシリアル番号の1つに対応し、発行済みシリアル番号ポイ

ンタ (I S N) 3 1 4 を発行された各命令について1つ前進させることで、新しく発行された各命令に A - リングに記述項を作る。1つのマシンサイクル中に複数の命令が発行されることがあるため、各サイクルで1個以上の A - ビットを設定し、 I S N を各サイクルで1つ以上の A - リングロケーションで前進させることができる。

I S U 2 0 0 による「命令ディスパッチ」は、実行のための (シリアル番号を割り当てられた) 命令の起動と D F B 6 2 への送信である。ディスパッチされたすべての命令は発行された命令でもある

が、発行された命令のすべてがディスパッチされるわけではない。「ディスパッチ」されたすべての命令は、「1」に相当してセットされた A - リング 3 1 2 中の割り当てシリアル番号に対応し、その命令がマシン中でアクティブであることを示す (1 = アクティブ、0 = インアクティブまたは非起動) 。発行されたがディスパッチされない命令は既知の分岐または n o p タイプの命令に対応し、その命令について A - ビットが発行時にクリアされる (またはセットされない) 。これら既知の分岐と n o p タイプの命令は、実行のための実行ユニットを必要とせず、1つの段階で発行、実行、完了することができる。

b. 発行ステージ間のポインタ関連の命令発行およびポインタメンテナンス

図 1 1 は、ポインタ記憶メモリ領域 3 1 0 に記憶され、 A - リング 3 1 2 および M - リング 3 2 4 上のロケーションを指す数個のポインタ (I S N 、 N I S N 、 C S N 、 R R P 、 N M C S N 、 および P B S N) も示す。それぞれのポインタは、 A - リングのロケーション値 0 ~ 6 4 の 1 つを記憶するポインタ記憶ユニット 2 1 0 の構成要素である。ポインタの組合わせによっては、 A - リングまたは M - リングのロケーションを指し示すことがあるが、ロケーションを指し示すことのない組合わせもある。例えば、 N I S N は I S N と等しくなることはあり得ないが、装置によっては、 C S N = I S N = R R P と「同期」する。発行されたシリアル番号ポインタ (I S N) 3 1 4 および次発行のシリアル番号ポインタ (N I S N) 3 1 5 は、命令発行状態を保持し追跡すると共に、全体的に他のポインタの利点に制約を加える。コミット (委託) されたシリアル番号ポインタ (C

SN) 316は、他のポインタには特に規定されていないような命令の実行、完了、および非活動化に続く命令コミット

を追跡する。リタイア再生ポインタ(RRP) 317は、命令のリタイアを追跡し、リソースの再生を制御する。最速予測分岐命令ポインタ(PBSN) 318および非メモリコミットシリアル番号ポインタ(NMCSN) 319の2つの補助的なポインタを使用して、予測分岐命令やロード/ストア命令のような待ち時間の長い命令の実行をスケジュールし追跡する。ICRU301はCPU51内の多数の他のユニットに対して、それぞれのISN、NISN、CSN、RRP、およびNMCSNに現在値を与える。ICRUは、以後説明するように、PSU300内のウォッチポイント・ユニット308からPBSN318の値を受け取る。これらのポインタは、実施例のCPUではそれぞれ6ビットベクトルとして実施される。

発行されたシリアル番号ポインタ(ISN) 314は、常に最後に発行された命令のシリアル番号を指し示している。ICRUによって与えられたシリアル番号の1つが実際にISU200によって割り当てられたとき、命令が発行されたと考えられる。ISNおよびNISNは、ICRUにそのサイクルの間に実際に発行された命令の数を通知するISU200からのISSUE_VALID 信号に応じてインクリメントされる。ISNおよびNISNがインクリメントされるので、ポインタは効率良くアーリング312の周りに前進する。マシンサイクルごとにISNが前進できるポジションの最大数は、マシンのピーク発行速度によって決定されるが、ハードウェアを制御する別のソフトウェアによって、この最大数をサイクル当たりの命令の一層小さな所定の値にまで制限できる。例えば、典型的なCPUでは、ISNおよびNISNの前進は、マシンサイクル当たり4つの命令のシリアル番号(アーリングのロケーション)に制限されている。CPUが初期化されると、ISNは初期化されて0(ゼ

ロ) になり、別の命令が発行されると、ISNは新たに発行された命令の数によって前進する。

次発行のシリアル番号ポインタ (N I S N) 3 1 5 は、常に I S N + 1 を指しているため (モジュロ A - リング長) 、 I S N が 63 の場合、 N I S N はゼロである。これは本質的に発行されるべき次の命令のシリアル番号である (すべての命令は予測された P C 順に発行される) 。独立したポインタの N I S N を提供することは便利であるが、 N I S N は常に I S N よりも 1 だけ大きいため、実施例の中には I S N だけを使用しているものもある。 C P U が初期化されると、 N I S N は初期化されて 1 (いち) になり、別の命令が発行されると、 N I S N は新たに発行された命令の数によって前進する。 I S N および N I S N の初期化と前進とは、 I S U 2 0 0 からの ISSUE_VALID 信号に応じて、 I S N / N I S N 前進論理ユニット 3 2 1 によって実行される。

I C R U 3 0 1 は、命令が発行されたときに割り当てられる 4 つの利用可能な命令のシリアル番号を I S U 2 0 0 に与える。適切な 4 つの命令のシリアル番号の決定は、 I S N 3 1 4 および N I S N 3 1 5 の現在値に基づいて、ポインタ制御論理ユニット 3 1 1 内の次命令シリアル番号割当てロジック 3 2 2 の中で行われる。 R R P 3 1 7 は、 C U がフルの場合、すなわちすべてのシリアル番号がすでに割り当てられ再生できない場合、シリアル番号の割当てを制限できる。 I C R U 3 0 1 は、 SN_AVAIL 信号を I S U 2 0 0 に送ることによってシリアル番号が利用可能になる場合、発行された命令に割り当てられるべき (4 つの) シリアル番号の次のシリーズを I S U 2 0 0 に通知する。

C P U 5 1 のある実施例では、 SN_AVAIL 信号は発行ウィンドウのスロット 0 ~ 3 において最大 4 つの命令に割り当てられるシリアル

番号を識別する 4 つの 7 ビット信号 (SN_SLOT_0, SN_SLOT_1, SN_SLOT_2, および SN_SLOT_3) から構成する。 6 つのビットはシリアル番号を表し、 7 番目のビットはシリアル番号妥当性ビットである。そのシリアル番号に対する妥当性ビットがセットされていない場合、命令のシリアル番号は C P U 5 1 内では依然としてアクティブであり、 I S U は使用してはならない。 6 4 ビットの A - リング 3 1 2 がサポートするすべての 6 4 ビットの命令が C P U 5 1 の中でアクティブである場合、妥当性ビットはセットされないことがある。シリアル番号が有効な場

合は I S U 2 0 0 は 1 つの命令しか発行できないので、I S U は次の命令を発行する前に、以前発行した命令が実行およびリタイアするまで待機しなければならないことがある。命令の実行およびリタイアについて、以下に説明する。

図 1 2 は、Aーリング・ビットセット／クリア・ロジック (A S C L) 3 1 3 の構造を含む状態制御論理ユニット 3 0 9 をより詳細に示している。A S C L 3 1 3 は、セット論理ユニットおよびクリア論理ユニット 3 3 1, 3 3 3 とそれぞれ関連したアドレスデコード論理ユニット 3 3 2, 3 3 4 ならびにセット論理ユニット 3 3 1 およびクリア論理ユニット 3 3 3 から構成する。セット論理ユニット 3 3 1 は、I S U 2 0 0 から ISSUE_VALID 信号を、またアドレスデコード論理ユニット 3 3 2 からデコードされた N I S N ポインタアドレスを受け取る。Aーリングのセット論理ユニット 3 3 1 は次に、I S U によって実際に発行された命令の数に一致する Aービットを Aーリング書込みポート 3 4 2 を介して「1」にセットする。命令が非活性化されるおよび／または C P U がリセットされると、Aービットはクリアされる。書込みポート 3 4 2 の数を選択して、各サイクルごとに活性化（または非活性化）される命令のメーカをサポートする。Mーリング 3 2 4 に関して図 1 2 で示した他の構造

を、長いレイテンシー（待ち時間）命令の積極的なスケジューリングおよび命令のコミットおよびリタイアと共に以下に説明する。

ある実施例では、ISSUE_VALID 信号は I S U 2 0 0 から受け取った 4 ビットのベクトルであり、このベクトルではポジションビット i での表明（位置付けられたときに「1」すなわち高い信号レベルにセットされたビット）は、 i 番目の命令発行ウィンドウスロットにおける命令が発行されていることを示している。I C R U 3 0 1 は、前に説明した SN_SLOT_ i 信号によって、どのシリアル番号がどのスロットに関係しているかを知っている。命令は連続したプログラムの順番に発行されるので、4 つの命令発行 C P U の ISSUE_VALID 信号に対して可能な状態は 5 つしかない ("0000", "0001", "0011", "0111", および "1111")。I C R U は、「ノップタイプ」の命令信号 (ISSUE_NOP) も I S U から受け取る。この信号は、説明したように命令がノップタイプの命令であると I C R U に通知する。

C P U 5 1 のある実施例では、ISSUE_NOP 信号は 4 ビット信号であり、ポジションビット i での表明ではスロット内の i 番目の命令が「ノーオペ」クラスの命令であることを示す。積極的なローカル／ストア命令のスケジューリングを実行する場合、ISSUE_MEM 信号は 4 ビット信号であり、ポジションビット i での表明ではスロット内の i 番目の命令がメモリ参照命令であることを示している。

一般的に、ノーオペ命令は分岐命令のような制御転送命令、またはライト A S R、ライト F P R S、またはライト P I L 命令のいずれかである。ライト A S R、ライト F P R S、およびライト P I L 命令は、D F B 6 2 ではなく、対応する制御レジスタ・ユニット 8 0 0 内で実行され、これにより N O P タイプの命令として処理される。従って、その対応するISSUE_NOP 信号は「1」にセットされ、このため A ービットは「0」にセットされる。Tcc 命令はCCが不明

の場合は発行されない。Tcc 命令は、CCが既知で誤りである場合、NOP 命令として発行される。Tcc 命令は、他のケースではISSUE_NOP=0 で発行される。これ等の命令は、実行するために D F B 6 2 実行ユニットを必要とせず、一般に発行されたときと同じマシンサイクル内に終了する。対応する命令が、「分岐」(br) 命令のような「ノップ」クラスの命令の場合、ISSUE_NOP 信号は「1」にセットされる。ISSUE_NOP 信号が「1」にセットされると、対応する A ービットは「0」にセットされる、すなわち命令は即座に非活性化されまたコミット可能となる。(命令コミットおよびコミット時の A ービットの変更については、以下に説明する。)

A ーリングの 6 4 のアクティビティビットは、C P U 内に共存する命令の活性を示す。A ービットは、使用される前のサイクルごとにセットされまたクリアされるので、初期化する必要はない。ISSUE_VALID, ISSUE_NOP, およびNISN信号に従って、A ービットはセットされる。

分岐命令は発行時にクリアされる A ービットを備えているので、追加のステップが実行されて、C S N 3 1 6 が投機的に実行された分岐命令を越えて前進しないようにする(以後の説明を参照のこと)。典型的な実施例では、分岐命令が発行されるときに、分岐条件コードがウォッチポイント・ユニット 3 0 4 に記憶さ

れる。ウォッチポイント・ユニット304は、命令の実行をモニタし、投機的に発行された分岐命令が終了するとき、ウォッチポイントは実行結果を必要な分岐条件コードと比較して、投機的な発行を確認する。条件コードを比較することによって、投機的な実行が正確に予測されたかについての決定が行われる。投機的な実行が正確でない場合、(バックアップおよび/またはバックステップのような)CPU51の回復手段が起動されてその実行を取り消し、これによりアービ

ットがクリアされているにもかかわらず、CSNが誤予測された分岐を過ぎて前進しないようにする。そうしないと回復を妨げる可能性がある。誤って発行された命令シーケンスを命令コミットすることを避けるために、マシン回復手段を十分早く起動するための特別な注意を取らなければならない。評価に先立って、保留の命令自体が誤って予測された命令シーケンスがコミットされないようにする。ウォッチポイントおよびCPU51の回復手段については、この明細書の他の場所でさらに詳細に記載する。

命令の状態を追跡し詳細な状態を維持するための命令タグの使用法を含む本発明による方法の実施例の態様を、図13～図15のフローチャートで説明する。図13は、状態を追跡し詳細な状態を維持するための命令タグの使用法の実施例の概略フローチャートである。図14は、命令の状態を追跡して詳細な状態を維持するための本発明による方法の実施例を示す概略フローチャートである。図15は、本発明の実施例によるアクティブ命令リングおよびメモリ命令リングに状態情報を書き込みまた維持する方法のフローチャートである。

3. レジスタのリネーミング

レジスタの依存性を取り除いてISU200がマシンサイクル当たり一層多くの命令を発行できるようにするために、CPU51は発行段階の間にレジスタのリネーミングを実行する。これは、1994年10月11日にシェバナウ(Shebanow)等に授与された米国特許第5,355,457号に記載されている方法、および/または「マイクロプロセサの物理レジスタの使用を調整するための方法と装置(METHOD AND APPARATUS FOR COORDINATING THE USE OF PHYSICAL REGISTERS IN A MICROPROCESSOR)」という名称の同時継続出願番号第08/388,364号に記載されている方

法、および／またはジョンソン (Johnson) の論文「スーパースカラー・マイクロプロセサの設計 (Superscalar Microprocessor Design)」のページ48～55に記載されている方法と同じ方法で実行できる。

レジスタのリネーミングを実行するために図8を参照すると、DFB62には、FXRFRN604、FPRFRN605、およびFSR/CCRFRN606のCCRFRN610が含まれている。前に簡単に説明したように、FXRFRN604には固定小数点データを記憶するための物理固定の小数点レジスタが含まれ、FPRFRN605には浮動小数点データを記憶するための物理浮動の小数点レジスタが含まれ、CCRFRN610には整数と浮動小数点条件のコードデータを記憶するための物理整数（固定小数点）および浮動小数点条件のコードレジスタ (XICCおよびFCGS) が含まれている。FXRFRN604、FPRFRN605、およびCCRFRN610は、それぞれ図16に示すように構成できる。

図16に示すように、レジスタファイルおよびリネームユニット (RFRN) 612には、物理レジスタを含む物理レジスタファイル614が含まれる。RFRNは、物理レジスタファイル614内の物理レジスタへの命令によって規定される論理レジスタおよび／またはアーキテクト・レジスタをマッピングするリネームマッピング・ロジック613も含む。例えば、SPARC-V9命令では、FXRFRN604のリネームマッピング論理は、80の論理128の物理変換固定小数点レジスタに構築された32のマッピングを実行でき、32のアーキテクトまたは論理的な単精度および倍精度の浮動小数点レジスタは、2組の64の単精度 (32ビット) のリネームされた浮動小数点レジスタ (奇数および偶数) にマッピングされ、5つのアーキテクトまたは論理条件コードレジスタ (xicc, fcc0, fc

c1, fcc2, およびfcc3) は32のリネームされた条件コードレジスタにマッピングする。アーキテクト論理変換マッピングは次の事実に基づいている、すなわちSPARC-V9の固定小数点命令は、図17に示す制御レジスタ・ファイル800のC

WPレジスタに記憶されたカレントウィンドウ・ポインタ（CWP）に従って、論理レジスタにマッピングするアーキテクトレジスタを規定する。

RFRNの制御ロジック613は、各発行段階の間にBRB59のフェッチレジスタから命令（F_INSTRs_BRP）を受け取る。これに応答して、制御ロジック613は命令をデコードし、どの命令がこの特定のRFRNのレジスタファイル614からのソースおよび／または宛先として物理レジスタを必要とするかを求める。

リネームマッピング・ロジック615は、先ず始めに、レジスタファイルの中で現在マッピングされた物理レジスタへの命令によって指定されたそれぞれの論理またはアーキテクト・ソースレジスタをマッピングし、TAGS_R信号の中で適当な実行ユニット（FPU 600, FXU 601, FXAGU 602, またはLSU 603）のリザーベーションステーションへの対応する物理レジスタタグを与える。それぞれの物理レジスタタグに対して、リネームマッピング・ロジックは、マッピングされた物理ソースレジスタ内のデータがまだソースとして利用可能かどうかを識別するデータ有効（DV）ビットを有する。DVビットは、リネームマッピング・ロジック615によってDV_R信号の中の適当な実行ユニットのリザーベーションステーションに与えられる。さらに、データがDVビットによって示されるように利用可能な場合、それはDATA_R信号内のマッピングされた物理ソースレジスタによって提供される。

それぞれのRFRN（すなわち、FXRFRN604, FPRFRN605, およびCCRFRN610）に対して、FREELI

STユニット700は、RFRNのレジスタファイル614のフリーな物理レジスタのすべてのリストを含む。フリーすなわち利用可能なレジスタとは、リタイアしていない命令によって現在使用されていないレジスタのことである。これらの利用可能な物理レジスタは、各サイクルごとにFREELISTユニット700からのFREE_REGS 信号によって識別される。

従って、論理またはアーキテクト・ソースレジスタがマッピングされた後、リネームマッピング・ロジック615は次にFREE_REGS 信号によって識別された利

用可能なすなわちフリーな物理レジスタに対する命令が指定したそれぞれの論理またはアーキテクト宛先レジスタをマッピングし、対応するDVビットをセットしてそのレジスタ内のデータはまだ利用可能でないことを示す。リネームマッピング・ロジック615は次に、新しくマッピングされた物理宛先レジスタに対して物理レジスタタグを与え、また対応するDVビットをTAGS_R信号およびDV_R信号の適当な実行ユニットのリザーベーションステーションに与える。

さらにリネーム・ロジック615は、P S U 3 0 0の再生フォーマットユニット(R R F) 3 0 2に旧論理またはアーキテクトされたレジスタマッピングの物理レジスタマッピング変換を提供する。前にマッピングした物理宛先レジスタに対する論理すなわちアーキテクトされたレジスタタグをLOG_OPHYS_TAGS信号のR R F 3 0 2に送ることによって、リネームマッピング・ロジック615はそれを行う。それぞれのR F R N 6 1 2からのLOG_OPHYS_TAGS信号は、これらの信号がその特定のR F R Nから来ることを示すためのフィールドを含んでいる。

R R F 3 0 2をさらに詳細に図18に示す。このR R Fはデータ記憶構造体366を含んでいる。最大64の発行されたがまだリタ

イアしていない命令にシリアル番号を割り当てることができる典型的なC P U 51では、データ記憶構造体366は64のアドレス可能な記憶素子またはエントリを含んでいる。各記憶素子は、命令のシリアル番号の1つに対応している。さらに、データ記憶構造体366の各記憶素子は、F P 論理旧物理変換マップフィールドまたはF X 論理旧物理変換マップフィールドを、またF C C 論理旧物理変換マップフィールドまたはX I C C 論理旧物理変換マップフィールドを含んでいる。各種の構造体はR R F 3 0 2を実行するために使用できる。ある典型的なR R Fは、64レジスタ、29ビットの4つのバンクのインターリーブされたR A Mとして実行され、そこではR R Fへの書込みサイクルの間、フィールドはパックまたはバンドルされ、バックステップの間の読み出しでアンパックまたはアンバンドルされる。

R R Fの制御ロジック365は、D F B 6 2からLOG_OPHYS_TAGS信号を受け取り、F X R F R N 5 0 4からの浮動小数点レジスタタグ、F P R F R N 6 0 5か

らの固定小数点レジスタタグ、およびCCRF RN 610からのXICCタグおよびFCCタグをどれが含んでいるかを決定する。制御ロジック365は、PSU 300のICRU 301からNIS Nポインタを、またISU 200からISSUE_VALID 信号も受け取る。NIS NのポインタとISSUE_VALID 信号に応答して、制御ロジック365は、LOG_OPHYS_TAGS信号を受け取る発行された命令のシリアル番号を決定する。

典型的なCPU 51では、命令は浮動小数点とFCCレジスタとを同時にまたは固定小数点とXICCレジスタとを同時に変更できるにすぎないが、固定小数点、浮動小数点、XICC、およびFCCレジスタを同時に変更できない。従って、固定小数点または浮動小数点の論理および旧物理レジスタタグをLOG_OPHYS_TAGS信号の中

で受け取るそれぞれの命令に対して、制御ロジック365はこれらのタグ(FP_TAGS または FX_TAGS) をシリアル番号を付けて発行された命令に対応する記憶素子のFP論理物理変換マップフィールドまたはFX論理物理変換マップフィールドに書き込む。同様に、FCCまたはXICCの論理および旧物理レジスタタグを受け取るそれぞれの命令に対して、制御ロジック365はFCC論理旧物理変換タグまたはXICC論理旧物理変換タグを、シリアル番号を付けて発行された命令に対応する記憶素子のFCC論理物理変換マップフィールドに書き込む。従って、命令が浮動小数点とFCCレジスタとの両方または固定小数点とXICCレジスタとの両方を必要とする場合、この命令のLOG_OPHYS_TAGS信号に含まれた論理および旧物理レジスタタグは同じ記憶素子に書き込まれる。しかしながら、RRF 302は固定小数点、浮動小数点、XICC、およびFCCレジスタを同時に変更する命令に対する論理旧物理変換マッピングを記憶するように構成できることは、当業者は理解されよう。

従って、RRF 302は、必要な場合、以前の論理物理変換マッピングまたは関係を再構成できる連続的な履歴台帳として機能する。例えば、SN="X"に対応する命令を取り消すことが必要になる場合、RRFの中でSN="X"をインデックスとして用いて、論理および旧物理レジスタタグを識別する。次に、これ等のタグは

適当な R F R N に渡され、SN="X"命令を実行する直前に存在した物理レジスタマッピングに対する論理レジスタマッピングを回復し、この新たにマッピングされた物理レジスタはフリーリストに戻される。これによりレジスタのリネーミングを効果的に逆にでき、レジスタの状態は命令を実行する直前に存在した状態に戻される。数個の命令を取り消す必要がある場合、同じ手順が逆にしたステップごとに、I S N から始めてSN="X"までデクリメントして加えられる。このプロセス

は詳細に後述する。

図19は、論理ソースレジスタL41およびL42、および宛先レジスタL47を備えた加算命令用のレジスタリネーミングの例である。前述に従って説明すると、論理ソースレジスタL41およびL42はリネームマッピング・ロジック615によって物理レジスタP50およびP52にマッピングされる。次に、論理宛先レジスタL47は、F R E E L I S T ユニット700が提供したフリーな物理レジスタP99に再マッピングされ、論理宛先レジスタL47用の論理レジスタタグおよび以前マッピングされた（旧）物理レジスタP78が加算命令に割り当てられたシリアル番号（SN=13）を用いてマッピングするためR R F に与えられる。この例から明らかなように、各「物理」レジスタタグは、（1）フリーリスト700、（2）F X R F R N 6 0 4、F P R F R N 6 0 5、またはC C R F R N 6 1 0 のリネームマッピング・ロジック、（3）R R F 3 0 2 の1つにまた1つだけに現れる。

4. リザーベーションステーションへのディスパッチ

図8をさらに参照すると、発行段階の間に、F P U 6 0 0、F X U 6 0 1、F X A G U 6 0 2、およびL S U 6 0 3 のリザーベーションステーションはS R B 5 9 から F _ I N S T s _ B R P 命令を受け取る。これに応答して、それぞれのリザーベーションステーションは、これ等の命令からオペコードと即値データとを抽出する。

それぞれのF P U 6 0 0、F X U 6 0 1、F X A G U 6 0 2、およびL S U 6 0 3 のリザーベーションステーションは、発行段階の間に発行された命令に割り当てられたシリアル番号およびチェックポイント番号を受け取る。これ等の番号はI S B 6 2 から受け取ったシリアル番号およびCHKPT_N 信号が提供する。

浮動小数点データを含む数学上の命令およびリード／ライト命令

を実行するために、FPU600のリザーベーションステーションは、発行段階の間に、FSR607のFPRFRN605およびCCRFRN610から、使用可能なデータ、データ有効ビット、およびFP物理レジスタタグおよびCC物理レジスタタグ (FP_DATA_R, FP_DV_R, FP_TAGS_R, CC_DATA_R, CC_DVR, CC_TAGS_R) を受け取ることができる。同様に、FXU601は、固定小数点を含む数学上の命令、プログラム制御命令、およびリード／ライト命令を実行するために、発行段階の間に、FXRFRN604およびCCRFRN610から、ISB2DFBバスに対する制御レジスタ・ファイル800からのデータ、使用可能なデータ、データ有効ビット、およびFX物理レジスタタグおよびCC物理レジスタタグを受け取ることができる。ロード／ストア命令および固定小数点非乗算／除算の演算命令用のアドレス発生動作を実行するために、FXAGU602は、FXRFRN604および／またはCCRFRN610からデータならびにFX物理レジスタタグおよびCC物理レジスタタグ (FX_DATA_R, FX_DV_R, FX_TAGS_R, CC_DATA_R, CC_DV_R, CCTAGS_R) を受け取ることができる。しかしながら、前に説明したように、プログラム制御、乗算／除算、および制御レジスタのリード／ライト命令を実行するために、FXAGU602はFXU601と同様に構成することができる。この場合FXAGU602はFXRFRN604およびCCRFRN610から適当な使用可能なデータおよび物理レジスタタグも受け取ることができる。さらに、ロード／ストア命令を実行するために、LSU603は発行段階の間にFPRFRN605および／またはFXRFRN604およびFSR／CCRFRN606のFSRレジスタの内容から、使用可能なデータならびにFP物理レジスタタグおよびFX物理レジスタタグ (FX_DATA_R, FX_DV_R, FX_TAGS_R, FP_DV_R, FP_TAGS_R) を受け取

ることができる。

そして、リザーベーションステーションのエントリがENTRIES_AVAIL 信号によって識別されたように使用可能な発行されたそれぞれの命令に対して、そのリザー

ーションステーションは識別されたエントリに抽出されたオペコード、シリアル番号、チェックポイント番号、物理宛先レジスタタグおよび／またはソースレジスタタグ、およびその命令に対して使用可能なデータを配置する。さらに、受け取ったデータ有効ビットによって識別されたように、データがすでに使用可能であるそれぞれの物理ソースレジスタタグについては、リザベーションステーションはリザベーションステーション内の別のデータ有効ビットをセットして、レジスタタグに関係するデータが使用可能であることを示す。また、命令が即値データを含む場合は、そのデータは即座に使用可能でありまたリザベーションステーションのエントリに記憶されるので、データ有効ビットもセットされる。

C. 命令の実行と完了

再度図6を参照する。発行段階の間に、P C ロジック 1 0 6 はフェッチレジスタ 1 1 0 からフェッチ命令 (F_INSTS_BRP) を受け取り、それをデコードする。P C ロジック 1 0 6 は I S U 2 0 0 からISSUE_VALID 信号も受け取り、実際に発行されたどの命令をデコードしているかを求める。1つのマシンサイクル内に発行され、実行と完了が予測できるプログラム制御命令については、このP C ロジックはプログラムの流れを予測し、その予測に基づいてF P C、A P C、およびN A P Cの値を計算する。その他のすべての命令については、P C ロジックは現在のマシンサイクルに対するF P C 値に基づいて次のマシンサイクル用のF P C 値と、ISSUE_VALID 信号が示すように現在のマシンサイクルの間にいくつの命令が発行される

かを計算する。これらの命令についても同様に、P C ロジックは以前のマシンサイクルのA P C 値に基づいてA P C 値およびN A P C 値を計算し、また現在のマシンサイクル内にISSUED_VALID信号が示すいくつの命令が発行されるかを計算する。

例えば、SPARC-V9 BPcc, Bicc, BPr, FBfcc および FBPfcc 命令のように、発行された命令に条件付き分岐命令がある場合、P C ロジック 1 0 6 は分岐を取るように予測するかまたは取らないように予測するかを決定するために、これ等の命令に対応するB R P ファイルをデコードする。分岐を取るように予測した場

合、PCロジック106は命令から置換値を抽出し、その値を使用して次のマシンサイクル用の新しいFPC値、APC値、およびNAPC値を計算する。分岐を取らない場合は、FPC値、APC値、およびNAPC値は、プログラムの流れに変更なく標準的に計算される。

しかし、発行された命令の中に、SPARC-V9 JMWPL[rd=0]命令のようなリターンタイプのジャンプおよびリンク命令がある場合、PCロジック106はリターン予測スタック105から予測されたJMWPLターゲットPC (P_JMWPL_PC) 値をポップオフするためにPOP信号を出力する。PCロジック106はこのターゲットPC値を使用して、新しいFPC値、APC値、およびNAPC値形成する。サブルーチンから復帰するためにJMWPL[rd=0]命令を使用するが、PCロジック106は、SPARC-V9 JMWPL[rd=15]命令のようなCALL命令すなわちコールタイプのジャンプおよびリンク命令が発行されたと判断したときはいつでも、PUSH信号を出力してリターン予測スタック105にP_JMWPL_PC 信号を送る。リターン予測スタック105に送られたこのP_JMWPL_PC 信号は、8だけインクリメントする命令用のAPC値である。さらに、CALL命令すなわちJMWPL[rd=15]命令が発行されるときはいつでも、P_JMWPL_PC 信号がリ

ターン予測スタック105に送られるので、PCロジック106はRETURN命令が発行されるときは常に、P_JMWPL_PC 信号がリターン予測スタック105からポップオフされることも確認しなければならない。

さらに、発行された命令には、SPARC-V9 BPA, BA, FBA および FBPA 命令のように無条件常時分岐命令が含まれることがある。この場合、PCロジック106はこれらの命令から置換値を抽出し、新しいFPC値、APC値、およびNAPC値を計算するために使用する。

PCロジック106は、条件分岐命令、常時分岐命令、およびJMWPL[rd=0]命令に対して、DFB62からの補足データを待つ必要がないので、これ等のタイプの命令は同じ段階で発行、実行、また完了することができる。さらに、発行された条件分岐が実施されるまたは実施されないことが予測され、また予測されたターゲットFPC値がJMWPL[rd=0]命令について計算されるので、続いてフェッチさ

れた命令は投機的にフェッチされこのため投機的に実行される。

さらに、発行された命令の中には、コールタイプのJMPL[rd=15]命令のようなCALL命令およびRETURN命令、ならびにJMPL[rd=0]命令以外のJMPL命令のように他の種類のプログラム制御命令がある。この場合、PCロジック106は、FPC値を計算し実行段階の間にその値をDFB2BRBバスに乗せるためにDFB62を待つ。図8を参照すると、FPC値は固定小数点データであるので、FXU601はFPC値を計算する。そして、完了段階の間に、PCロジック106はこの値を取り込み、新しいFPC値、APC値、およびNAPC値を形成する。

後でより詳細に説明するように、発行されたいくつかの命令がSPARC-V9 DONE命令またはRETRY命令のようなトラップハンドリング

・ルーチン命令からリターンするとき、PCロジック106はトラップPC (TPC) 値またはトラップNPC (TNPC) 値を、図7に示す制御レジスタファイル800が提供したRD_TPC_TNPC信号から受け取る。図17を参照すると、制御レジスタファイル800のトラップスタック815は、ISU200がRETRY命令またはDONE命令が発行されたことを示すRETRY_DONE_IS信号を出力するとき、これ等の信号を提供する。PCロジック106は次にトラップスタック814から受け取ったTPC値またはTNPC値を使用して、新しいFPC値、APC値、およびNAPC値を形成する。

他の発行された命令については、PCロジック106は発行段階の間に前述のPC値およびNPC値をインクリメントして、次のフェッチ段階用の新しいPC値およびNPC値発生する。

図8を参照する。発行段階の間に発行された命令は、必要なすべてのデータが対応するリザベーションステーションのエントリに配置された場合のみ、FPU600、FXU601、FXAGU602、およびLSU603を実行できる。他の命令よりも後で発行された命令用のデータがより早く使用可能になるので、これ等の命令は以前発行されたよりも前に実行され完了されることがある。

前述したように、物理ソースレジスタ用のデータ有効ビットが使用可能と指示

しない場合は、物理ソースレジスタのソースデータは使用できないことを、リザベーションステーションは知っている。従って、リザベーションステーションは、データが使用可能な場合、FXRFRN604、FPRFRN605、およびCCRFRN610から転送された、適当な使用可能なデータ、データ有効ビット、および物理レジスタタグ (FX_DATA_F, FX_DV_F, FX_TAGS_F, FP_DATA_F, FP_DV_F, FP_TAGS_F, CC_DATA_F, CC_DV_F, CC_TAGS_F) を受け取りモニタする。リザベーションステーションは、対応す

る転送された物理レジスタタグがリザベーションステーションのエントリに記憶された物理レジスタタグに適合する場合は、特定のレジスタ用に意図された転送データを受け取る。

すべてのデータ依存性がある命令に適合した場合 (すなわち、すべての必要データが使用可能になる場合)、その特定の命令用にリザベーションステーションのエントリに記憶された命令を用いて、実行ユニットは次にその命令を実行する。実行ユニット600~603が実行する命令のタイプについては、リザベーションステーションの記憶エントリに関して前に説明してある。

生成したデータがその命令用のエントリのリザベーションステーションに記憶された物理宛先レジスタタグによって識別された物理レジスタに記憶されたとき、実行された命令は完了する。完了する間に、実行ユニット600~603はERR_STAT信号のエラーおよび完了状態情報もPSU300に送る。

さらに、図7を参照すると、ISU200はフェッチレジスタ100から受け取った命令をデコードして、発行された命令のいずれかが図17に示す制御レジスタ801~814のリード/ライト命令であるかどうかを決定する。制御レジスタ801~814はCPU51の全体的な管理のもとで使用され、SPARC-V9のアーキテクチャマニュアルの中で説明されたタイプの特権付きのステートレジスタ、特権なしのステートレジスタ、および補助ステートレジスタを含むことができる。

SPARC-V9 RDPR 命令およびRDASR 命令のような発行された制御レジスタのリード命令については、レジスタ801~814の1つを識別して読み込まれること

になっていることを示すRD/WR_CNTRL 信号を I S U 2 0 0 は出力する。その応答として、制御レジスタ・ファイル 8 0 0 の制御ロジック 8 1 6 は、RD/WR_CNTRL 信号が識別し

た制御レジスタの内容 (RD_CNTRL_REG) をウォッチポイントユニット 3 0 4 に出力する。

後述するように、C P U 5 1 は制御レジスタ 8 0 1 ~ 8 1 4 のリード命令に対してチェックポイントおよび対応するウォッチポイントを作るので、I S U 2 0 0 はDO_WATCHPT信号およびDO_CHKPT信号を発生する。DO_CHKPT信号は、チェックポイントを作成しこのチェックポイントに対するチェックポイント番号を含むべきであることを示している。DO_WATCHPT信号は、DO_CHKPT信号が提供したチェックポイント番号にあるウォッチポイントユニット 3 0 4 によって形成されるべきであることを示している。ウォッチポイントユニット 3 0 4 は、それぞれが 1 6 のチェックポイント番号の 1 つに対応する 1 6 のアドレス可能な記憶素子を有する。従って、DO_WATCHPT信号およびDO_CHKPT信号に応答して、ウォッチポイントユニット 3 0 4 は、DO_CHKPT信号の中で提供されたチェックポイント番号に対応する記憶素子の中で読み取られる制御レジスタの内容を記憶する。

リード制御レジスタ命令は、さらに実行を続けるために、F X U 6 0 0 のリザーベーションステーションにも提供される。従って、命令のチェックポイント番号はその命令用のリザーベーションステーションのエントリの中に記憶される。そして、F X U 6 0 1 がリード制御レジスタ命令を実行するとき、それは命令のチェックポイント番号をウォッチポイントユニット 3 0 4 に出力する。その応答で、ウォッチポイントユニットは F X U 6 0 0 より前に読み取られた制御レジスタのデータを出力する。F X U 6 0 0 は次にそれを提供し、また固定小数点レジスタに記憶するために対応する物理宛先レジスタタグを F X R F R N 6 0 4 に与える。従って、C P U 5 1 は制御レジスタの読み込みを行うために同期される必要はない。

制御レジスタ (SPARC-V9 WRPR およびWRASRのような) に書込み

動作を行うためには、F X U 6 0 1 はリザーベーションステーションにその命令用に記憶された命令データを実行してデータを D F B 2 B R B バスに出力する。図 1 7 に示すように、そのデータは次に制御レジスタ・ファイル 8 0 0 に送られる。I S U 2 0 0 は、制御レジスタへの書込み命令が発行されまたどのレジスタにこの書込み命令が行われるかを示す RD/WR_CNTRL 信号を発生しているので、RD/WR/UPDATE ロジック 8 1 6 はデータを適切に制御レジスタに書き込むことができる。

前述したように、典型的な C P U 5 1 は固定小数点用にレジスタウィンドウを使用している。レジスタウィンドウの動作は、SPARC-V9 のアーキテクチャマニュアルに詳細に記載されている。従って、I S U 2 0 0 はいつレジスタウィンドウの制御命令が発行されたかを決定する。そのように行う際に、I S U 2 0 0 はウィンドウ制御命令が発行されたことを示すと共に特定の種類の行うべきレジスタウィンドウ動作を識別する WIN_CNTRL 信号を発生する。これ等の信号は、図 1 7 に示すように、制御レジスタ・ファイル 8 0 0 に渡される。その応答として、RD/WR/UPDATE ロジック 8 1 6 は次に、WIN_CNTRL 信号が示すレジスタウィンドウの更新動作を実行する。

D. 命令実行段階における P S U の参加

ひとたび命令が D F B 6 2 内の適当な実行ユニットにディスパッチされると、I C R U 3 0 1 は命令が完了するのを待ち、それぞれの命令のシリアル番号についてエラーおよび状態情報が D F B 6 2 および P S U 3 0 0 の間の実行データ転送バスをわたって到来することをモニタする。実行段階における P S U の参加は、実行状態情報を受信すること限定されている。命令がエラーなしで実行を完了し誤った予測がなかった場合、この命令は非活性化される。しかしながら、実行の例外が発生するかまたは投機的な実行が誤って予

測された場合、この命令は非活性化されないかまたは C P U 5 1 は回復手順を初期化する。A-リングの状態情報は更新されて、正確または不正確な投機的実行および例外的または例外なしの実行を反映する。命令が正しく予測され実行が例外なしに予測された場合、I C R U 3 0 1 はシリアル番号に関連する A-リング

のビットをクリアしてその命令を非活性化する。

典型的なCPU 51においては、DFB 62は浮動小数点機能ユニット(FPU) 600、固定小数点機能ユニット(FXU) 601、固定小数点/アドレス発生機能ユニット(FXAGU) 602、およびロード/ストア機能ユニット(LSU) 603、ならびにFPRFRN 605、FXRFRN 604、CCRF R N 606、およびFSR 607を含む。典型的なCPUでは、それぞれのDFB機能ユニットはマシンサイクル当たり2つの命令まで処理できるので、命令の実行完了はマシンサイクル当たり最大8つの命令に(ハードウェアの制約によって)制限される。CPUのある実施例では、FPU 600は2つの命令を同時に処理できるが、1つの命令の結果しか出力できない。そのためその実施例では、命令の完了は7つの命令に制限されている。しかしながら、2つの出力を実行できない理由はない。図8はDFB 62の中の主な機能ユニットと、DFB 62およびISB 61間を通過する主な信号の機能的ブロック図を示す。

ディスパッチされた命令がDFB 62の中で実行を完了すると、命令識別エラーおよび他の状態情報(ERR_STAT)は、実行データ転送バス上のICRU 301(および他のユニット)を含むPSU 300に同報通信される。データと状態は各命令に対して受け取られるが、順序を乱して受け取られることがある。各命令用の実行状態情報信号には、例えば、実行中にエラーが発生したかどうか、エラーのタイプ、投機的な実行が正しく予測された場合の評価用の条件コード、シリアル番号、命令のチェックポイント(マシンが故障した場合およびバックアップを必要とする場合、マシンがバックアップするチェックポイント)、および命令には制御レジスタに対するリードまたはライト命令が含まれているかどうかの情報を含むことができる。

同報通信命令のシリアル番号は、図12のアドレスデコード・ロジック334によってA-リング312にインデックスするために使用される。A-リング上のA-リングライトポート341, 342(およびCSN/R RP前進ロジック323用のリードポート343)の数は、プロセッサのピーク実行帯域幅に一致するように選択するのが好ましい。特定の命令がエラーの原因になる場合、A-

リングクリアロジック333はこの特定の命令のシリアル番号に相当するA-ビットに「0」を書き込み、この命令は以後非活性化されたと考えられる。しかしながら、命令がエラーを伴って終了した場合、A-リングの対応するアクティブビットはクリアされないため、この命令はなおアクティブと考えられ、例外条件またはエラー条件から回復するために、プロセサを回復させる別のステップが用いられる。結局、CSNは、エラー処理命令に先行するロケーションのあるスロットを指すISNに追いつくことになる。CSNはそのA-ビットがまだセットされているため、エラー処理命令のシリアル番号を通過できない。そして、実行トラップが行われるとき、エラー処理命令のA-ビットは「1」で上書きされ、CSNおよびRRPが前進することを可能にする。

トラップのタイプを識別するテイクントラップ(TT)フィールドはトラップスタックからポップオフされデコードされて、このトラップが実行トラップ(etrap)であるかどうかを決定する。この

実行トラップは次に「DONE」命令または「RETRY」命令を発行することによって処理される。実行トラップをうまく処理できた場合、トラップハンドラによってETRAP_CLEAR信号が現れ、A-ビットをクリアする。同様のロジックがM-リング324のクリアM-ビットに送られる。

データ信号および状態信号は、それぞれの実行発行スロットにおける命令のシリアル番号を提供すると共に、どの例外が一時的に早いかを定めることを複数の命令が誤るような場合に、PSU300が例外処理のプライオリティを決定するために使用される。(例外の優先順位については、プライオリティロジックおよびステートマシンPLSM307に関連してさらに説明する。)典型的なプロセサにおいては、シリアル番号はエラーおよび状態情報よりも1サイクル早く到来するため、エラー情報の到来の前にアドレスデコード・ロジックユニット334がシリアル番号をデコードすることが可能であり、このためアドレスデコード・ロジックの中で一層の遅延を招くことなく、ビットをクリアできる場合(エラーがない場合)またはビットをクリアできない場合(エラーがある場合)がある。エラーまたは他の例外が検出された場合、この明細書の中で後述するように、こ

のエラーまたは例外を処理するための適切なステップが実施される。

E. 命令コミット

これまで説明した構造と方法は、典型的な実施例におけるAーリング312と64の命令のアドレス可能なロケーションと同数の命令の発行および実行状態を追跡する構造および方法を提供している。CPUのリソースを回復する構造と方法が欠けていても、CPU51によって命令に割り当てられた、例えばシリアル番号および関連するAービットを含むリソースは、例えば命令が完了し非活性化さ

れていても、その命令に対して割り当てられたままであろう。Aーリングのデータ構造に関連する付加的なポインタを付け加えて、またCPU51内の関連する命令の状態の変化に応じてそのポインタを移動させることによって、特に非活性化と命令をコミットすることができ、またリソースをリタイアさせて別の命令によって後で使用するために再生できるようになる。この方法では、リソースは新たに発行された命令が連続的に使用できるようになる。

コミットシリアル番号ポインタ(CSN)316は、最新のコミットされた命令を指し示す。「コミットされた命令」とは、エラーなしで実行が完了した命令と定義されるため、それは発行から実行を通して進行し、取り消される必要があるまた取り消されることがないいかなる投機的に発行された予測中央転送(ブランチ)命令よりも前に発行されている。コミットされた命令を取り消す必要はない。定義により、コミット命令シリアル番号(CSN)ポインタ316より前のすべての命令は、エラーなしで実行が完了しており、取り消される必要はない。CSNがマシンサイクルごとに進む量は、マシンの最大命令コミット速度を設定する。完了しておりかつ非活性化された命令は、Aーリングにおけるこれ等の命令用のシリアル番号に向かってまたは超えてCSNが前進するまでコミットされるとは考えられない。

ISN314について前述したように、CSN316が前進できる最大のポジション数は、ソフトウェア、ハードウェア、または他の制御によってサイクル当たりの命令のより小さな最大数に制限できる。例えば、ISNの前進をマシンサイクル当たり4つの命令に制限する発明のある実施例では、CSNはハードウェ

アに課せられた制限によってサイクル当たり8つに制限される。さらに、CSNはISNに等しくなることもあるが、A-リングの周りではISN

を超えて前進することはできない。(A-リングの構造のため、CSNは厳密には数的にISNより小さく、等しく、または大きくなることができる。)CPUが初期化される時、CSNは0(ゼロ)に初期化され、次に命令の状態の変化および、以後説明するように、所定のCSN前進ルールに従って前進する。

アクティブビットはアクティブ命令リングの中でクリアされるので、CSNは所定のルールに従って前進して、実行が完了している命令を「コミット」する。一般に、CSNのポインタは、すでに実行を完了した命令より遅れることがある(すなわち、非活性化される)。CSN/RRP前進ロジックユニット323は、CSNポインタおよびRRPポインタのポジションを前進させることに責任を負う。各マシンサイクルで、CSN/RRPロジックユニット323はISNを通らずに、完了した非活性化された命令(A-リングにおいて、アクティビティビット=「0」)を超えてCSNを前進させようとする。据置きトラップが生じた場合を除いて、CSNはA-リング312のA-ビットのセットされた条件またはクリアされた条件および実行された(例えば、ソフトウェア、ファームウェア、および/またはハードウェア)所定のルールに基づく場合のみ前進する。CSN/RRPロジックユニットはCSN用の適切なシリアル番号のアドレスを決定するために、A-リング自体の状態を問い合わせ、ISNを注意し続ける必要があるだけである。据置きトラップが得られてうまく処理された場合、PLSM307はDEFERRED_TRAP信号を、CSNを1つのシリアル番号のロケーションだけ前進させるCSN/RRP前進ロジック制御・ユニット323に送り、このためCSNと後のRRPが例外の原因となる命令を通して前進できるようになる。本発明のある実施例では、CSNの前進は所定の最大命令コミット速度(マシンサイクル当たり8命令)

によって制限される。この速度はCSNポインタを移動させるために必要なハードウェアを制限する希望に基づいて選択される。

原則として、CSNはISNと等しくなるまで前進できる。ISN=CSNの場合、CPUは空である、すなわちすべての発行された命令はコミットされている。実際、ロジックを単純化できまた必要なハードウェアを少なくできるので、CSNの最大の前進量を（マシンサイクル当たり（4命令発行マシンについては）8つに）制限することは有利である。例えば、マシンサイクル当たり4命令発行CPUでは、CSNの前進量はサイクル当たり8つのシリアル番号に制限されている。CSNの前進量を支配するルールは、以下のように要約できる。（1）CSNの前進はISNまでである（両端を含む）。（2）CSNの前進は1マシンサイクル当たり8命令のシリアル番号以下である。（3）前進はまだコミットできないアクティブ命令に相当するAーリングの最初の「1」までである（両端を除く）。さらに厳密にいうと、CSNは一般に次の関係に基づいて前進する（下記の例外に関する修正を参照のこと）。すなわち、 $CSN = \min(CSN+4, ISN, \text{slot_A})$ 、ここで、 $A(CSN+1), \dots, A(\text{slot_A})$ はすべて「0」であり、 $A(\text{slot_A}+1)=1$ である。この式で「min」は最小関数であり、「slot_A」はAーリング312のAービットのロケーションのことであり、また $A(\text{slot_A})$ はそのAービットロケーションのビットの状態（例えば、「1」または「0」）である。据置きトラップが削除されまた未完成の浮動小数点オペレーションの例外またはデータブレークポイントの例外が検出されたように受け取られたとき、CSNを前進させるルールは修正される。未完成の浮動小数点の例外およびデータブレークポイントの例外は、DFB62からの状態信号（ERR_STAT）を実行することによって識別される。これ等の例外が発生しトラップが処理されると、CSNは

1だけ前進する、すなわち $CSN = CSN + 1$ となり、CSNは例外の原因となった命令を通して前進する。PSU300は未完成の浮動小数点オペレーションおよびデータブレークポイントに特に関心を持っており、Aービットが処理されたトラップに対してクリアされない場合、CPUは満杯となり機能停止するため、未完成の浮動小数点オペレーションおよびデータブレークポイントの両方はポイントの特別の処理を必要とすることがある。

CSNの前進を「maximum(8, ISN, the next A-bit = 1)」に制限することに

よって、CSNを適切なポジションに移動させるための単純なハードウェアの回路を実現できる。例えば、A-リング312は8つの8ビットレジスタとして実現できる。これ等8つの8ビットレジスタのA-リングのリードポート343において、CSNの周囲のレジスタが読み込まれ、これ等の読み込まれたレジスタの内容は2つの別個の8ビットバスに送られる。これ等のバスを単一の16ビットバスに連結して、次に0ビットから8ビットまでシフトすることによって、古いCSNを前進させるための8ビット関心ウィンドウを形成する（CSNの最大の前進量は8）。「最初の1を見つける」回路は、CSNが前進する量を見つけることになる。

ICRU301はまたコミットシリアル番号（CSN）信号および発行転送番号（ISN）をチェックポイントユニット303に送ることによって、現在のコミットシリアル番号（CSN）316のチェックポイントユニット303に通知し、CSNより小さいシリアル番号を有する任意の割り当てられたチェックポイントを解放できるようにする。厳密な数学的な順序（すなわち、シリアル番号）ではなく円形のA-リング上のロケーションに比例するもので、CSNは円形のA-リング上のロケーションに比例する。チェックポ

イントについては、この明細書の各所でより詳細に説明している。CPU51のある実施例では、本願に記載したCSNならびに他のポインタは、アドレス可能なA-リングのビットロケーションと一致する命令のシリアル番号をエンコードする6ビットのベクタである。

F. 命令のリタイアメントとリソースの回復

CPUの割り当て可能なリソースはプロバイダによって回復される。補助ポインタは、命令がリタイアするときに解放されるリソースに対して命令のリタイアメントおよびリソースの再生を追跡および制御するために提供される。リソース再生ポインタ（RRP）317は、最新のリタイアした命令を指し示し、CSNを追跡する。「リタイアした命令」とは、実行が完了し、CPU51によって非活性化され、またコミットされた命令のことであり、次の命令の発行の間に、次の使用のために再生された関連するマシンのすべてのリソースを有している。再

生されたマシンのリソースには、リネームされたおよび／または再マッピングされた論理レジスタ、アーキテクチャルレジスタ、および物理レジスタが含まれる。これ等のレジスタは、命令のリタイア時に開放されこのため次の命令が発行される間に再割り当てに使用可能とされた命令、命令のシリアル番号（SN）、チェックポイント、およびウォッチポイントに割り当てられている。マシンサイクルごとにRRPが前進する量は、そのマシンの最大命令リタイアメント速度を設定する。CPUが初期化されると、RRPは0（ゼロ）に初期化され、命令の状態の変化および後述する所定のRRPの前進ルールに従って前進する。チェックポイントとウォッチポイントの割り当ておよび割り当て解除については、以下に詳細に説明する。

命令がコミットされると、リソース再生ポインタ（RRP）31

7は、命令をリタイアするためにCSN316の後ろに前進するため、マシンのリソースを解放することができまた再利用のために再生できる。それぞれのマシンサイクルごとに、AーリングのAービットの状態とCSNを超えない所定のRRPの前進ルールとに基づいて、RRP317はコミットされた命令を超えて前進しようとする。非コミット命令に割り当てられたマシンのリソースは、たとえ非活性化された場合でも、命令が取り消される必要がないことがはっきりするまで開放されないので、RRPはCSNを通過できない。非活性化された命令はまだ投機的であるので、命令の完了（非活性化）だけでは命令コミットまたはリタイアメントに対して十分ではない。コミットされた命令のみがリタイアできる。RRP317はレジスタのリソースを再生できる次の命令を指し示している。Aーリング312、CSN316、およびCSN／RRP前進ロジック328の中で実行できる所定のルールのセットされた条件またはクリアされた条件のみに基づいて、RRP317は前進する。

ISNおよびCSNの場合のように、RRPがそれぞれのマシンサイクルごとに前進できるポジションの最大数は、他のソフトウェア、ハードウェア、または他の制御によって、サイクル当たりの命令のより小さな最大数に制限される。本発明のある実施例では、RRPの前進量は、RRPのポインタを移動させるに必

要なハードウェアを制限したいという希望に基づいて選択された所定の最大命令リタイアメント速度によって制限される。例えば、典型的なサイクル当たり4発行タイプのCPUの実施例では、RRPの前進とこれによるリソースの回復は、マシンサイクル当たり最大4命令に制限される。これは典型的なマシンの最大命令発行速度に等しいため、CPU51の性能を制限するものではない。RRPを通常の発行速度よりも遅く前進するように制限すべきではない。さらに厳密に言

うと、RRPは次の関係、すなわち、「 $RRP + \min(RRP + 4, CSN)$ 」に基づいて前進する。原則的に、この方法でリソースの回復を制限することは、本質的なものではなく、RRPはCSNまで前進できる。ハードウェア設計のトレードオフのために、典型的なCPU51に制限が課せられた。CSNを所定の命令数だけ前進させるに必要なハードウェアは、RRPを同じ命令数だけ進めるに必要なハードウェアよりも大規模になる。従って、ハードウェアを希望のレベルまで小さくするために、性能に有害な影響がほとんどまたは全くないようにして、RRPをCSNよりも遅く前進させる。

1. 命令のコミットと命令のリタイアメント時のRRFとリネームマップの更新

図18を参照する。リソースリカバリは命令のシリアル番号を開放するので、そのシリアル番号を再び使用することができ、その結果、いくつかのレジスタを開放することになり、そのレジスタはフリーリストユニット700内のフリーリストに戻すことができる。命令がコミットされると、この命令はICRU301が発生したCOMMIT_VALID信号によって識別される。このCOMMIT_VALID信号に応答して、RRF302の記憶ユニット366のコミットされた命令と一致するエントリは、再利用のため制御ロジック365によって開放される。

2. 命令のリタイアメント時のRRFの読取り

命令がリタイアするとき（ICRU301の説明を参照のこと）およびマシンのバックアップ時（バックトラックの説明を参照のこと）に、RRFは読み取られる。ある命令がリタイアする場合、RRPポインタが示すように、制御ロジック365は記憶ユニット366を制御して、フリーリストユニット700に、FREE_TAG信号の形式で、リタイアした命令に一致する物理レジスタタグを出力する

。フリーリストユニットは次に、この物理レジスタタグを空きの物理レジスタタグのリストの中に入れる。

3. ポインタを使用する精確なステートメンテナンス

Aーリング312およびその関連するロジックは、発行された各命令の状態に関する正確なマシンサイクルの現在の情報を提供する。ISNとCSNのポインタは、発行されたアクティブな命令の状態を追跡する。CSNが特定の命令に向かって前進する場合のみ、CPU51は、その命令をコミットできまたフリーなリソースをその命令の実行と関連付けることができると確信することができ、これにより精確な状態を保存できる。ISNはエントリを通過するとき、発行された命令にAービット ($A=1$) をセットするAーリングの周りを移動する。CSNは、コミットするために完了した命令 ($A=0$) を探すISNに続く。NMCSNは、完了した (非活性化された) メモリ命令 ($M=0$) を探すISNに続く。RRPはAーリング312の周りのCSNの後に従い、コミットされた命令のリソースを再生する。積極的なロード/ストア命令のスケジューリングを説明する場合、その説明にはどのようにNMCSNが完了した (非活性化された) メモリ命令 ($M=0$) を探すISNに続くかが記載されることになる。

CSNはAービットがセットされている命令へは進まない。(NMCSNは、Mービットがメモリ参照命令の発行時にクリアされるので、メモリ参照命令を通過して前進する。) 正確な状態を維持できるポインタを移動させるための他のルールは、以下の通りである。すなわち、(1) CSNはISNを通過して前進できない ($ISN \geq CSN$)。 (2) CSNはNMCSNを通過して前進できない ($NMCSN \geq CSN$)。 (3) CSNはPBSNを通過して前進できない ($PBSN \geq CSN$)。 (4) NMCSNはISNへ前進できない ($ISN \geq NMCSN$)

。(5) PBSNはISNへ前進できない ($ISN \geq PBSN$)。 (6) RRPはCSNを通過して前進できない ($CSN \geq RRP$)。 (7) NISNは常に $ISN+1$ である (加算はモジュロリング長である)。 (8) ISNはRRPまたはCSNに追いつくことはできない (最大限のマシン条件を示す)。好ましい実行方法において任意に実行できる他のルールとして、(9) CSNおよびNMCSNは1クロック

サイクルで最大8前進できる。(10)RRPはマシンの1クロックサイクルで最大4前進できる。記号「 \geq 」は、より大きいまたは等しい関係は、厳密に数学的に等しいまたは等しくないということではなく、Aーリング構造の周りのラップに関して決定されることを示す。

ISNポインタ、CSNポインタ、およびRRPポインタに関して、次のように観察できる。すなわち、第1に、CSNとISNとの間のシリアル番号を持つ命令は、マシン内で未決着のアクティブな命令に相当する。これ等の命令は投機的であり、取り消す必要がある。第2に、CSNとRRPとの間の命令はすでにコミットされたリタイアしてない命令に相当するもので、RRPによるリタイアメントを単に待っている。これ等のコミットされた命令は取り消されることはない。第3に、CSNはまだ発行されていない命令の結果をコミットできないので、ISNを通り越すことはできない。第4に、定義により $CSN = ISN = RRP$ の場合に「シンク」信号が到着するので、マシンで発行されたすべての(ISN)命令はコミットされ(CSN)リタイアされる(RRP)。第5に、ISNは概念的に円形のAーリングにおいてRRPを超えることはできず、またRRPとISNとの間のシリアル番号は命令の発行に使用できる。 $ISN = RRP - 1$ の場合(モジュロAーリング長)、すべての命令のシリアル番号は割り当てられ、そしてRRPが前進して

別の命令が発行される前にマシンのリソースを再利用するために再生できるまで、マシンはストール(待機)しなければならない。無論、CPU51はストールの前に発行された命令の実行、非活性化、完了、およびコミットを継続する。最後に、3つのポインタ間の相対的な関係は、 $RRP \leq CSN \leq ISN$ であり、ここで「 \leq 」のオペレータは円形のAーリング(モジュロAーリング長)内の順序付けを示すものであり、厳密な数学的な関係を意味するものではない。図20は命令の発行、非活性化、コミット、およびリタイアメントを含む精確な状態を維持するための、本発明による方法の実施例の概略フローチャートである。

III. 積極的な長レイテンシ(長待ち時間)(ロード/ストア)命令のスケジューリング

命令のコミットとリタイアメントを追跡し予定する各種のポインタを使用する本発明による方法の説明は、すべての命令のタイプに関係するものであり、現在の投機的な順序なし (out-of-order) のプロセサでは多くの利点を提供する。多くの現在のプロセサは、ロード命令およびストア命令のみが外部メモリを参照し、一方他のすべての命令またはオペレーションはプロセサ内のレジスタを参照するような「ロードストア」アーキテクチャとして設計されてきたことを認識することによって、さらに別の利点を得ることができる。

実際に、ロード命令とストア命令は、この明細書ではレイテンシ (待ち時間) の長い命令と呼ぶより一般的なクラスの命令に属しており、このレイテンシの長い命令は完了するには数個のマシンサイクルを必要とし、ロード命令とストア命令が2つのタイプである外部メモリ参照命令を含む。内部レジスタを参照する命令は、一般に完了するためにはより少ないまたは単に1つのマシンサイクル

しか必要としない。レイテンシの長い命令も、割り込みなしに複数のマシンサイクルで実行するように設計された命令である「原子」命令を含んでいる。

レイテンシとは、命令が発行されてからその命令の実行が完了するまでの間の経過時間のことである。ロードストアアーキテクチャに対する命令のセットは好ましい、というのはそれがレイテンシの長いメモリ参照 (メモリへのロードストア参照) を、CPU 51 の内部レジスタを参照するレイテンシの短い論理演算または算術演算 (加算、乗算、除算、論理比較、など) からデカップリングするためである。さらに、ロードストアタイプのアーキテクチャで設計された最も新しいCPU 51 は、このタイプの命令セットから発生した増大するレジスタへの要求事項を支援できる多数の汎用目的のレジスタを有している。

ロード命令とストア命令は両方ともアーキテクチャの状態を修正する。順序なしに実行するプロセサでは、正確な状態を危うくすることなく高いロード/ストア帯域幅を維持するために、ロード命令とストア命令とを効果的に予定できることが重要である。正確な状態を維持しながらレイテンシの長い命令を積極的に予定する、命令の状態を追跡するための基本的な構造と方法への改良が提供される。PSU 300 のICRU 301 内の構造と方法は、性能を改良するために後で

説明するが、D F B 6 2 の、特にロード／ストアユニット（L S U）6 0 3 内の構造と方法に協力する。

A. メモリ命令状態情報記憶

本発明による構造と方法は、長レイテンシ（長待ち時間）命令を区別する他の信号と、長レイテンシ命令状態情報を記憶する記憶領域と、命令状態を追跡するポインタを供給する。第21図は、メモリ参照命令（例えばロード／ストア命令）に代表される長レイテ

ンシ命令状態情報を記憶するメモリ命令リング（Mリング）324 と前述の能動命令リング311 との関係を示す概念図である。Mリング324 は一般にAリング311と同じ構造と全体特性を持っているが、Mリング状態ビットはAリング状態ビットとは異なる組の規則に従ってセットされる。特にMリング324 は、命令と命令形式が発行時に能動（ビットが“1”にセットされる）または非能動（ビットが“0”にクリアされる）として処理される規則を設定する。

第10図に示すようにISU200が命令を発行する時には、前述のISSUE_NOP 及びISSUE_VALID と共にメモリ参照命令と非メモリ参照命令とを区別する命令形式信号を供給する。ロード命令、ストア命令、原子命令順序及び他のメモリ参照命令を含め、積極的にスケジュールされることによって利益を生む特定クラスの長レイテンシ命令を示すようにISSUE_MEM を発生させることもできる。

CPU51 の一種の実施例では、ISSUE_MEM はビットでのアサーション（“1”）が i 番目の命令がメモリ関連であることを示す4ビットベクトル（各ビットは命令発行ウィンドウ中で発行可能な命令の1個に対応）となっている。ISSUE_MEM は、第12図に示すようにMリング324 にビットをセットするためにMリングセットロジック7335によって用いられる。Aリング312 内にはアドレスデコードロジック338, 336が設けられている。ISSUE_MEM は発行された命令が記憶動作に関係するか否かを示す。ISU200からのISSUE_MEM がビット i でアサートされて同じ命令スロットに対応するISSUE_VALID がアサートされると、メモリ命令リング（Mリング）324 はメモリ命令リング（Mリング）324 にはセットされない。命令がメモリに関連している時にMビットをセットしないことにより、CSN がクリアさ

れたAリングビットへ進むように非メモリコミテッド命令シリアルn番号(NMCSN)ポインタ(後述)がクリヤされたMビットへ進み

、非メモリ参照命令によって進むことを防止されるように、発行時にメモリ関連命令を有効に「非活性化」することができる。ISSUE_VALID が“1”の時、即ちメモリ関連であろうとなかろうと発行された各命令に対しては、Mビットは必ず“0”または“1”として書かれなければならない。記憶動作のMビットをセットしないことによりNMCSNを進めるために記憶関連動作を「無視」することが容易になる。

命令がエラーなく完了した時には、Aリング用にクリヤされた時と同じようにMビットはMリングクリヤロジック337によってクリヤされる。発行時にクリヤされたMビットを持っているメモリ参照命令は、命令の非能動化後もクリヤされたままである。Aリング312と同じように、Mリングビットは実行トラップのためにクリヤされる。Mリング324状態情報はメモリ関連命令を追跡し積極的にスケジュールするのに用いられる。メモリ参照命令の発行状態、コミテッド状態、リタイヤ状態を含むすべての状態は、Aリング311とそれに関連するポインタによって維持される。

AリングとMリングの前述の類似性からして、当業者はAリングとMリングの機能はリングのアドレス可能な場所が複数のビットを含んでいる単一の円形リングのような単一のデータ構造内で発効できることにお気づきであろう。このような複数ビットのリングでは、命令活性(能動または非能動)と命令形式(メモリ、分岐、NOP(ノップ)など)の両方共複数ビットにコード化できる。更に、現在別のAリングとMリングのデータ構造に供給されているのと同じ情報を記憶するために、他のコード化スケジュールが提供できよう。以上の説明からして、当業者は、マルチビット書式サポートしNMCSNを収容するためのリングセットとクリヤロジックの変更の仕方、マルチビット円形リング用のCSN/RRPロジックの変更の仕方、能

動と非能動との差異化とメモリ命令の差異化のためにセットには“1”クリヤに

は“0”とする以外に異なるロジック状態または記号を使用できることを御理解されることであろう。

B. 長レイテンシ命令を追跡しスケジュールするNMCSN ポインタとPBSNポインタ

Mリング324 にMビットをセットする構造と方法については以上に説明した。本発明の構造と方法では、長レイテンシ命令、特に外部メモリを参照するロード／ストア命令をCPU51 が有効且つ積極的にスケジュールすることができるようにするため、Mリング324 に関連して予測分岐命令シリアル番号 (PBSN) 318 とポインタと非メモリコミテッドシリアル番号 (NMCSN) ポインタ319 が供給使用される。NMCSN とPBSNは前述のように両方共ICRU301 内のポインタレジスタ313 に記憶される。

NMCSN319は最後の「コミテッド」命令を支持する (前述) が、命令発行時にメモリ参照命令用のMビットがクリヤされるので、NMCSN を進めるためにすべてのメモリアクセス命令は有効に無視され、従ってNMCSN は更に高速に進む。ポインタCSN316が進むだけで命令が実際にコミットされる点に御注目頂きたい。後述するようにNMCSN はロード／ストアユニット603 によってだけ使用される。命令がエラーなく完了し、それ以前のすべての命令がエラーなく完了している場合に命令をコミットすることができる。メモリ参照命令が未だ能動であってもNMCSN はメモリ参照命令を追い越して進むことができる。この方法はメモリを参照しない他の長レイテンシ命令形式を積極的にスケジュールするのにも使用できる。CPU が初期設定されるとNMCSN は0 (ゼロ) に初期設定される。

PBSN318 は一番初期の (一番古い) 未解決予測分岐命令のシリアル番号である。この実施例ではPBSNはウォッチポイントユニット304 によって決定される。(ウォッチポイントユニット304 については本明細書の他の箇所に詳述。) 未解決分岐命令とは、発行され実行されたことが確認される以前に理論上発行され、実行開始され実行完了されている可能性のある命令 (及びその命令から生じる命令順序) のことを言う。換言すれば、命令が発行された時には処理の流れの中にその命令を実行する条件が未だ生じてはいないのである

。本発明の構造及び方法では、未解決予測分岐命令が存在しない時にはPBSNは必ずISN と等しくなる。PBSNはNMCSN 進行ロジックでNMCSN 進行のバリアとして用いられる。NMCSN はポインタPBSNまたはISN を追い越して進むことはできず、メモリ命令が存在しない状況では、ポインタNMCSN はCSN に等しい。CPU が初期設定されるとPBSNは0（ゼロ）に初期設定される。

C. NMCSN の進行

NMCSN319はCSN316と対になってメモリ関連動作を行う。CSN を進めるAリングに対応してNMCSN を進めるMリングがある。Mリングはラップの周囲のMビット64個から成る。即ち、典型的には64ビットのMリングで、Mビット#63に後続するビットはMビット#0である。Mビット=0とは、メモリ関連命令、例えばロード／ストア命令のような長レイテンシ命令のことである。

例示したCPU では、NMCSN ポインタを進める規則は次の示すようにCSN を進める規則に類似している。（1）NMCSN をISN まで進める（ISNを除く）。（2）NMCSN を1サイクルで8個以下の命令シリアル番号以下進める。（3）Mリング内の最初の“1”まで進める（この“1”を除く）。

浮動小数点ユニット(FPU)で「未完浮動小数点動作」例外が生ずるかまたはロード／ストアユニット(LSU)から「データブレイクポイント例外」が検出された時にはNMCSN を進める規則が変更される

。これら2種類の場合には、前述したCSN と同様にNMCSN はシリアル番号1個だけ進められる。即ち、これらの状態では $CSN \leftarrow CSN + 1$ と $NMCSN \leftarrow NMCSN + 1$ になる。PSU300は特別のポインタ操作を必要とする未完浮動小数点動作とデータブレイクポイント動作に特に関係している。

MリングのインプリメンテーションはAリングのインプリメンテーションに似ている。例示したCPU ではNMCSN は事実上maximum(8, ISN、次のMビット=1)までしか進めないで、CSN と同様NMCSN を適当な位置へ進めるには簡単なハードウェア回路構成の設置計画をすればよい。(maximum(x, y, z)という表現は、パラメータ値またはx, y及びzという表現の中から最大値を選択するという意味である。) Mリング324 は8個の8ビットレジスタとして設置される

。これら8個の8ビットレジスタのMリング読取り口346では包囲しているNMCSNのレジスタが読取られ、読取り時のこれらのレジスタの内容が2本の別々の8ビットバスに送られる。これらのバスを連結して1本の16ビットバスとし、次にこのバスを0ビットから8ビットへ移動させて古いNMCSNを進める問題の8ビットウィンドウを作る。「最初の1を発見する」回路がNMCSNの進む量を与える。

NMCSNとPBSNの保守を正しく行えば、精確な状態を曖昧化しない実行のために選択し得るロード／ストア命令またはその他のメモリ参照命令を簡単に決定することができる。精確な状態を曖昧化しない実行のために選択し得るロード／ストア命令は、実行に「インレンジ」だと考えられる。(1)例外を発生する命令がアーキテクチャ状態を変えず、(2)故障命令以前のすべての命令がアーキテクチャ状態への変更を完了し、(3)故障命令以後のすべての命令がアーキテクチャ状態を変えていない時に精確な状態が維持さ

れることをもう一度述べておく。LSU603はまた、長レイテンシ命令のスケジュールに使用するため、ICPU301からのNMCSN319とウォッチポイントユニット304からのPBSN318を受信する。

D. データフローブロック内のロード／ストアユニット(LSU)

本発明の特徴のいくつかの設定は一般的な実行ユニットによって得られるものであるが、長レイテンシ命令の積極的スケジュールが利益を生み出すのは状態スケジュールに長レイテンシをサポートする特定の構造が存在する場合である。例えば、ロード／ストア命令と、他のメモリ参照命令を積極的にスケジュールする構造が提供される。局所的な記憶動作に参加しない他の実行ユニットにこの種のテストを設置する方法は、以上の開示からして当業者には自明であろう。

LSUは、SPARC-V9アーキテクチャマニュアルに定義されているリラックスメモリモデル(RMO)とトータルストアオーダリング(TSO)の両モードをサポートする。LSUは固定小数点と浮動小数点両方のロード／ストア命令に対して責任を持つ。LSUは各々がキャッシュへのサイクルである2個までのリクエストを作ることができる。精確な状態を維持するために必要な命令の順序立ては、プロセッサとキャッシュチップとの間で1組のプロトコル信号によって行われる。ISNは12項

目のリザーブステーションを内蔵している。LSU とデータキャッシュの間には分割されたトランザクションをサポートする 3 段階のパイプラインがある。第 1 段階では命令のアドレスと操作符号とシリアル番号がデータキャッシュへ送られる。理論上の実行に使用されるいくつかの制御ビットも送られる。第 2 段階では ISU からデータキャッシュへとストア命令のデータが送られる。第 2 段階ではまた、データキャッシュが次のサイクルで完了する命令のシリアル番号と有効ビットを LSU へ返送する。第 3 段階ではデー

タキャッシュが状態とロードデータを返送する。キャッシュミスの場合は、データキャッシュはパイプラインスロットが使用されていない間にデータを返送するか、そのデータのためにパイプラインスロットを開く信号をアサートする。

第 22 図は選択兼実行制御ロジックユニット 609 と、種々の命令及びデータ関連情報を記憶する命令／データ待ち行列(IDQ) 610 と、ISU200, FPU600, FXU601, FXAGU602からの信号を復号する復号&フィールド分離ロジック(SSFSL)／620 とを含むロード／ストアユニット(LSU)の機能ブロック図である。詳述すると、IDQ610はインレンジビットフィールド611 と、操作符号フィールド612 と、命令シリアル番号(SN) フィールド613 と、チェックポイント番号614 と、有効性ビットフィールド615 と、アトリビュート(ATR)フィールド616 と、レジスタタグ(TAGS) フィールド617 と、タグ有効フィールド(TV) 618 と、アドレスフィールド619 と、アドレス(AV) フィールド620 と、キャッシュデータフィールド621 を含んでいる。データキャッシュ52はIDQ610とアドレス及びデータを送受信する。

LSU603が命令を受信すると、デコード・フィールド分離ロジック622 は命令の直接データフィールドにアドレスを含めるか或いはFXAGU602がアドレスを計算しなければならないかを決める。デコードロジック622 がアドレスはFXAGU602によって計算されなければならないと決めると、アドレス有効ビットをクリヤする。しかし、命令の内部に直接データがあるという意味でアドレスが命令の内部に含まれると決めると、アドレス有効ビットをセットして、その命令用のアドレスデータが準備されて利用でき、その命令を実行することができると指示する。タグ信号がデータが有効か否かを指示するように、FPU と FXU から来るタグも同様に

操作される。命令がリザー

ブステーションまたは待ち行列610 に記憶されると、発行された有効信号が実際に実行されたことを示すまで有効信号がクリヤされる（アサートされない）。

LSU603が命令を実行すると、データキャッシュ52へ直接データを送る。LSU603がデータキャッシュ52からのデータを検索している時には、本発明の設置計画のデータは選択兼実行制御ロジックに入っており、このロジックはタグセンドを含む信号とデータが有効であることを示すデータ有効ビットを送出し、また図示したようにデータを送出するので、組合わせ信号は出て来ない。第22図にはFP_TAG S_DV_F信号とFP_DATA_F 信号が示されている。

選択実行制御ロジック609 もアクセス情報を得てこの情報をデータキャッシュ52に与える。この情報のあるものはトラップの型式がメモリ動作に関係しているトラップ情報であって、エラー検出に際してPSU300で用いるエラー信号（ERR_STAT）を出力する。

選択兼実行制御ロジックユニット（SECLU）609はLSU で実行する命令の選択に責任がある。このユニットはロード、ストア、アトミック動作などの順序立て強制を取り扱う。例えば、アドレス「X」への若いロードはアドレス「X」への古いストアを追い越すことはできない。この型のロードはストアのようなアトリビュートアレイでは「強い」とマーク付けされる。このユニットはストアの「順序立て」を開始し、すべての先行するロードとストアが完了するまでストアを開始することはできない。最後に、ストアのシリアル番号はインレンジでなければならない。SECLU609は、有効適格（即ちキャッシュによって現在処理されていない）であってすべての従属性を満し、且つ「弱い」動作（例えばロード）か「インレンジ」であってしかも前のキャッシュミスによって強制されていない命令だけをマスクするロジックを含んでいる。マスクされた命令は次に、マ

スクロジックロジック準備を満す命令の組から一番古い命令を選択するプリシードンスマトリックス623 によって処理される。

SECLU609もメモリアクセスの制御に責任がある。このユニットはキャッシュに

コマンドを送り、アクセス完了時にキャッシュ状態情報を受信する。また、キャッシュからの「ロード」データに適当な目的地レジスタタグを貼り、アクセスのアトリビュートのトラックをアトリビュートアレイ616に保存する。これらのアトリビュートはインレンジで、強／弱な（一般にストアとアトミックは「強く」、ロードは「弱い」）理論上のアクセスで、キャッシュへ送るに適格なものを含んでいる。

SSFSL620は命令操作符号（オペコード）と、命令シリアル番号と命令チェックポイントとレジスタタグを含む命令パケットをISU200から受信し、情報をデコードし、情報を記憶するためにIDO610で正しいスロットを選択し、情報を適当なフィールドに分離する。SNに関連する各シリアル番号項目とデータは待ち行列中の空のスロットに記憶される。

SECLU はポインタ値 (ISN, NISN, PBSN 及びNMCSN)もPSU300から受信する。CSN, PBSN 及びNMCSN は、命令シリアル番号がインレンジであるかの決定に特に関係する。SECLU609はアクセスが「インレンジ」であるか、即ち、シリアル番号が前述のようにCSN より大きくNMCSN 以下であるかを定めるロジックを含んでいる。インレンジ情報はアトリビュートアレイ616へ進められてSECLU内のインレンジファイル611に記憶される。SECLUはIDQに記憶されている項目をデータ返送状態及び／または与えられたアクセス用のキャッシュからのデータとマッチングするロジックも含んでいる。IDQ内のすべての有効項目はそのサイクル中インレンジであるかを見るためにモニタされ、アトリビュートアレイは必要に応じて更新される。待

ち行列項目が一旦インレンジになると、再使用されるまではインレンジのままである。

チェックポイントフィールド614は、待ち行列スライス中のアクセスに対応するチェックポイント番号を含んでいる。各サイクル毎に新しいチェックポイント番号がISB61から到着する。有効ビット待ち行列615は、メモリアクセス命令が実際に発行された場合にセットされる有効性ビットを保持する。この情報は命令発行サイクル中にISB61から来る。ISBによってアクセスが殺された場合にはビ

ットをリセットすることができる。アドレスフィールド618 はすべてのキャッシュアクセス用のアドレスを保持する。DFB62 アドレス発生ユニット (FXAGU) 602 からアドレス待ち行列にアドレスが到着する前に、アドレス番号がIDQ アドレスマッチロジックに到着する。制御レジスタの内容に応じてアトリビュートアレイ616 にアトリビュートをセットしてもよい。例えばプログラムオーダビットをセットすると、アトリビュートセクションには「強い」ビットがセットされる。

1. インレンジメモリ参照 (長レイテンシイ) 命令の識別

前述のように、SECLU609はICRU301 からのCSN とNMCSN 及びウォッチポイントユニット304 からのPBSNに応じて命令が「インレンジ」であるかを決めるロジックを含んでいる。このロジックはCSN とPBSN ($CSN \leq \text{第1窓} \leq PBSN$) の間のシリアル番号を持つ第1窓にある命令が分岐の誤予測に関係なく実行できるか識別する (但し、「 \leq 」の記号は厳密な算術順序よりもメモリ命令リング内で相対順序を示す)。第2のCSN とNMCSN ($CSN \leq \text{インレンジ} \leq NMCSN \leq PBSN$) の間のシリアル番号を持つ多分小型のウィンドウの命令は、分岐誤予測または実行例外とは無関係に実行するようにスケジュールされる。積極的な実行をスケジュールするためにDFB62 内のLSU200に識

別することができる「インレンジ」となっているのはメモリ参照命令である。第23図は厳密な状態を維持しつつロード/ストア命令を含む長レイテンシイ命令を積極的にスケジュールする本発明の方法の実施例を概略的に示すフローチャートである。

2. ベーシック構造への強化と長レイテンシイ (ロード/ストア) 命令の積極的スケジュール

本発明のベーシックな方法へと強化する際には、ロード/ストア命令とメモリを参照する他の命令を更に積極的にスケジュールできるようにしてNMCSN を進め性能を改善するために「非故障」として知られている所定の命令を無視するように規則を設定することができる。

IV. チェックポイント法

チェックポイントは既知の瞬間におけるマシン状態の「スナップショット」で

ある。チェックポイントは、誤予測分岐または実行例外が生じた場合に理論上の命令順序を元に戻すために前のマシン状態を素速く回復するための方法と構造を提供する。本発明による典型的なプロセッサでは、命令発行サイクルの間にチェックポイントが作られる。本発明の方法では、チェックポイントの回復に基づくマシン状態の回復はマシンの「バックアップ」と定義される。

本発明ではチェックポイントは数クラスの命令用に創られる。即ち、(1) 分岐のような予測プログラム制御転送命令と(2) 制御レジスタ値を変更するサイドエフェクトを持っているかも知れない命令である。

制御転送命令(分岐のような)は予測されプログラムカウンタ(PC)不連続を生ずるかも知れない。本発明の構造と方法を設定するCPU51では、プログラム制御命令(分岐またはリターン型のジャンプ及びリンク)が誤予測された場合に素速くマシンのバックアップ

を行うために、各予測プログラム制御命令に対してプロセッサがチェックポイントを作る。これらの状況でAPCとNAPCがセーブされないと、誤予測されたプログラム制御命令の再発行と再実行せずに正しいFPCとAPCとNAPCを再構成することは困難である。このようなプログラム制御命令の各々に対してチェックポイントが作られているので、予測できてしかも同時に未解決のままでいるプログラム制御命令の数は、マシン内に許容されるチェックポイントの最大数によって制限される。予測プログラム制御命令は前述したSPARC-V9 BPr, FBcc, FBPcc, Bcc, BPcc及びJMPL[rd=0]命令のような命令とすることができる。

プログラムフローを変える他の型の命令もチェックポイントで調べることができる。これらの命令にはCALL, JMPL, TCC, RETURN, DONE 及びRETRYのようなSPARC-V9命令が含まれる。

制御レジスタ値を変更するサイドエフェクトを持つことができる命令を含むマシン状態を変更する命令に対してもチェックポイントが創られる。ある制御レジスタをチェックポイントすることによってこの制御レジスタを理論上変更することができる。これらはSPARC-V9 WRPR 及びWRASR 命令のような制御レジスタファイル800内の制御レジスタへの書き込みを含み、SPARC-V9 SAVED, RESTORED及び

FLUSHW命令のようなレジスタウィンドウ制御命令も含む。しかし、性能を失わずにチェックポイントされる状態の量を少なくするためにすべての制御レジスタがチェックポイントされる訳ではない。

更に、例示CPU 51では、前述したように、制御レジスタのデータは後刻実行のためにFXU601で利用できるようにウォッチポイント304に記憶されるので、制御レジスタファイル800内の制御レジスタの読取りにはチェックポイントが必要である。これらはSPARC-V9 RDPR 及びRDASR 命令のような命令とすることができる。

更に、発行トラップを頻繁に起す命令もチェックポイントを必要とすることがある。これらには、SPARC-V9 SAVE 及びRESTORE 命令のようなウィンドウ制御命令と、例えばSPARC-V9 LDD/LDDA及び STD/STDA命令とすることができる設置計画されない命令が含まれる。

トラップも理論上エンターされうるが、そのように発行した時にはチェックポイントされる。従って、後に述べるように実行トラップ(etrp)または発行トラップを生ずる命令に先行して発行される命令にはチェックポイントを作ることができる。更に、理論上トラップが取られた場合に復帰が行えるように、トラップ操作ルーチンの前にチェックポイントが作られる。

理論的に制御することができないサイドエフェクトを持つ命令、即ち、例外を生ずることが稀で多量のチェックポイント情報を必要とする命令に対しては、効率上の理由から命令のチェックポイントを行わず、その代り発行前にマシンを強制的に同期化するように決定を行うことができる。即ち、理論的に制御できないサイドエフェクトを持つ命令を発行する前に、CPU をマシン同期とし、すべてのペンディングな命令がコミットしリタイヤされた($ISN=CSN=RRP$)とするのである。

どの命令をマシン同期命令とし、どの命令をチェックポイントするかを選択する時には重要な性能/設計の引き渡しが行われる。同期のための性能低下は、チェックポイントされたマシン状態を記憶するために要するチップ領域を含め、この種の命令の理論上の操作に関係する追加のロジックの複雑性とプロセッサ領域

と比較して評価される。

多くの異なる型の命令がレジスタ中のデータ値を変更する。従来のチェックポイント法ではレジスタ状態を回復するためにレジスタデ

ータ値をチェックポイントする。対照的に、本発明の方法をマシン設置する時には、実際のレジスタデータ値をチェックポイントせず、代りにレジスタ再命名マップをチェックポイントすることによってCPU51が以前のレジスタ状態を回復することができる。CPU51に記憶されているチェックポイントされた状態情報の量を大幅に減少できるので、「レジスタデータ値」ではなくて「レジスタ再命名マップ」をチェックポイントするのが有利である。

例示したマシンでは、コンカレント命令のサポーティング64により、16個までのチェックポイントを任意の時に利用できる。実施例ではISU200は発行サイクルにつき最大1個のチェックポイントの割り当てに制限されている（ハードウェアの考慮により）。しかし一般的には、1サイクルで1個または任意の複数個のチェックポイントを割り当てることができる。例示したCPU51では、同一のマシンサイクル中に発行できる命令の型と組み合わせを制限するように所定の規則が設置されている。例えば例示したCPUでは、1個のマシンサイクル中1個の制御転送命令だけを発行でき、従ってそのサイクルに状態をセーブするに要するチェックポイントは1個だけである。このチェックポイントの制限により、サイクル中に最大数（例えば4個）の命令を発行できない場合（発行ウィンドウで待期中の命令の型による）がある。更に多数または少数のチェックポイントを供給することもできるが、その数はマシンの中で共時的に目立つチェックポイントを必要とするナンバー命令のようなファクターに基づいて選択される。

実施例では、能動命令リング312が64個もの命令を同時にサポートする。後に説明するように、本発明の構造と方法は、ある種の命令型式のチェックポイントを行い、ISU200からのTIMEOUT_CHKPNNT信号に応じて所定の命令サイクルタイムアウト間隔で任意の命令型式

のチェックポイントを任意に行う。従って、サポートされるチェックポイントの

数は能動命令リング312によってサポートされる命令の数より大きくないことが必要で、この数は一般に命令の数より小さい。例示のCPU51では16個のチェックポイントを割り当てることができ、これらのチェックポイントの各々はチェックポイントを生じた命令がリタイヤされた後に回復されるマシン資源になっている。

例示のCPU51では、チップのルートを減らし性能を向上させるため、DO_CHKPT信号に応じてチェックポイントデータ(マシン状態)はチェックポイント記憶ユニット内のCPU51全体に局所的にセーブされる。しかし、チェックポイントされたデータを記憶消去されたチップにできることは自明である。

チェックポイント記憶ユニットの各々は16個のアドレス可能記憶素子または項目を持っている。各記憶項目は後述する16個のチェックポイント番号の中の1個に対応する。従って第6図を参照すると、DO_CHKPT信号に応じてAPCとNAPCレジスタ113及び114(CHKPT_PC及びCHKPT_NPC)の内容はPCロジック106によってPCチェックポイント記憶ユニット107に記憶される。同様に、制御レジスタの内容(CHKPT_CNTRL_REG)は、第17図に示すように制御ロジック816により制御レジスタチェックポイント記憶ユニット817に記憶される。更に第16図に示すように、FXRFRN604とCCRFRN610と、FPRFRN605の各々の再命名マップを表わすデータはチェックポイント記憶ユニット616に記憶され、フリーリストロジック701のフリーリスト(CHKPT_FREELIST)を表わすデータはフリーリストチェックポイント記憶ユニット702に記憶される。そして後述するようにトラップスタックユニット815の或る種の内容もチェックポイントされる。

ISU200はチェックポイントを作る時機の決定に責任がある。BRB59から受信した命令をデコードして、どれかがチェックポイントを要する前述の型であるか決定する。そして後述するように、所定数のマシンサイクルが行われた後にチェックポイントを作るようにPSU300がTIMEOUT_CHKPT信号によって指示した時、チェックポイントを作る時機を決定する。この結果、ISU200がチェックポイントを作ることを指示し、そのチェックポイントのチェックポイント番号を指定するDO_CHKPT信号を発生する。

第24図はPSU300内にあるチェックポイントユニット303の機能ブロック図である。ISU200によってDO_CHKPT信号を主張すると、1個または2個以上の命令に対してチェックポイントを作るようチェックポイント(CKPT)303に指示し、命令スロット0-3で発行されたどの命令がチェックポイントを要求したかを示す。少なくとも1個のフリーチェックポイントレジスタが利用できなければチェックポイントを割り当てることができないから、DKPT303はISU200へCHKPT_AVAIL信号を送らなければならない。CHKPTは少なくとも個のチェックポイントを割り当てることができることをISUに知らせ、チェックポイント番号が各命令と関係づけでき、ISU200によって作られたチェックポイント番号(CHKPTs)に合体でき、DFB62へ進められるようにチェックポイント番号をISUに知らせる。これらの信号はISU200によるDO_CHKPT信号の発行に先行する。チェックポイント番号を割り当てることのできない場合は、チェックポイント資源が利用できるようになるまでISUによる信号発行を待機させる。簡単に言えば、実行例外または誤予測から回復させるのに必要な場合だけ記憶されているチェックポイントを使用し、そうでない場合はチェックポイントの割り当てを解除し記憶スペースを解放して再利用するのである。(チェックポイントを含むマシン資源の割り当てと割り当て解除については後に詳述する。)後述する本発明のベーシッ

ク構造と方法へ強化するにも、所定数のマシンサイクルまたは所定数の命令が発行され、チェックポイントを作ることなく終結した時にデコードされた命令型式とは無関係にタイムアウトチェックポイントが作られる。

チェックポイントユニット303はチェックポイントのトラックを維持し、ISU200が指示した時にチェックポイントを割り当て、必要でなくなった時にチェックポイントを解放する。コミットッドシリアル番号がチェックポイントのシリアル番号($CSN > SN$) $i + 1$ よりも大きくなるとチェックポイント i をリタイヤさせることができるが、これは本発明の方法と構造によればマシン状態を任意の命令の境界に再記憶させることができるので、コミットされた命令よりも順序的に前にある命令までマシンを逆追跡する必要がないからである。PSU300はシステムの他の構成要素にもいつチェックポイントを割り当て解放するかを知らせるので、こ

これらの構成要素はリタイヤされたチェックポイントに使用されていた局所データ記憶領域を再使用することができる。チェックポイントユニット303 はまた、いくつかのチェックポイントが自由に割り当てできるかをISU200に知らせる。チェックポイントの使用については、後にバックアップ兼バックステップユニットに関連して詳述する。

チェックポイント303 は5種の主要機能を持っている。第1に、ISU200からのDO_CHKPT信号に応じて、チェックポイントレジスタユニット352 内のチェックポイントデータレジスタ構造 (CHKPNT_REG_i) のリストの保守によってISU200の要請する通りにチェックポイントの割り当てを制御する。制御転送信号が発行されると、DO_CHKPTは1個または2個以上のチェックポイントが作られるべきことと、それぞれのチェックポイント番号の位置 (例えば0-15) を指定する。本発明の構造と方法を強化する際には、後にベーシックなチ

ェックポイント法への強化において述べるように、TIMEOUT_CHKPT がチェックポイント303 によって発生されISU200へ送られて命令型式とは比較的無関係に所定のタイミングでチェックポイントを作る。このタイムアウトチェックポイントは、チェックポイントが作られなかった命令の効果を元に戻そうとするサイクルの最大数を決めて制限するために供給される。(チェックポイントできる命令型式が同じ命令ウィンドウから発行される時には、タイムアウトチェックポイントの形成は延期されるかキャンセルされる。)

第2に、チェックポイントは或る与えられたチェックポイント (n) を追い越して進んだCSN に基づくチェックポイントと仮に次に位置しているチェックポイント (n+1) をリタイヤ (クリヤ) させる。特定のチェックポイント番号を過ぎたCSN を決めるメカニズムは、CSN をチェックポイントレジスタに記憶されているシリアル番号と比較することである。チェックポイントレジスタに記憶されているシリアル番号とは、チェックポイントを形成した命令の次の命令のシリアル番号である。CSN がチェックポイントnとn+1を追い越して進んだ場合にだけ、例えばICRU301 からのISN とCSN に応じてチェックポイントnをリタイヤさせることができる。

第3に、チェックポイント303は、ISUによってチェックポイントが割り当てできるか否かとチェックポイントの位置番号を含め、ISU200へ送られる信号CHKPT_AVAILによって割り当てられる次のチェックポイントは何かをISU200に知らせる。第4に、チェックポイント303は割り当てられたチェックポイントに対応する命令のシリアル番号を記憶し、他のユニットによって要請された時に、ICRU301に対してチェックポイントに関連するSNとSN+1の両方を供給する。第5に、顕著な有効チェックポイントの中のどれを殺すかをバックトラックユニット305内のバックアップユニット402に示す命

令キルベクトル (KILL_VEC) を発生し、信号CHKPNT_SNとCHKPNT_SN_INCによってICRU301に対し、マシンのバックアップに関連してどのチェックポイントのリタイヤさせられるかを知らせることによって実行マシンのバックアップに参加する。(マシンのバックアップは後に詳述。)

A. チェックポイント割り当てレジスタ

チェックポイント割り付けレジスタ352の構造は表1に概略的に示してある。最初の6ビットは状態ビットである。ビットVはチェックポイントの有効性(割り付け)を示す。ビットCP(チェックポイントが追い越されている)はCSNがこのチェックポイントのSNを追い越したかを示し、QはCSN前進トラックを維持する5種の状態のうちの1つを示し、GT(より大きい)はCSNが割り付け時のチェックポイントのシリアル番号よりも大きいか否かを示す。ビットGTはCSNがいつこのチェックポイントを追い越したかを決定する。固有の機能性を得るためにビットVが「1」にセットされビットCP「0」にセットされるチェックポイント位置0以外では、ブータップ時にビットVとCPをゼロにセットすべきである。TSNフィールドはチェックポイントされたシリアル番号に後続している次のシリアル番号を示す。この設置計画では、チェックポイント番号はセーブされず、位置「i」用のチェックポイントデータ構造にVビットをセットするとチェックポイント「i」が割り付けられたことが示される。チェックポイントレジスタ(CHKPNT_REG_i)に可能な状態は表2に説明してある。例示したCPUでは、チェックポイントの割り付け中、ビットQは「000」か「001」にリセットされる。ビットQ

は A リング 312 の周囲の CSN の進行に基づく状態を変える。表 3 に Q ビットの状態と遷移を示す。

表 1. 典型的なチェックポイント割り付けレジスタ (CHKPNT_REG
_i) の構成

i	1-ビット	1-ビット	3-ビット	1-ビット	6-ビット
0	V	CP	Q	GT	NSN
1 ⋮ ⋮	V ⋮ ⋮	CP ⋮ ⋮	Q ⋮ ⋮	GT ⋮ ⋮	NSN ⋮ ⋮
14	V	CP	Q	GT	NSN
15	V	CP	Q	GT	NSN

表2. 典型的なチェックポイントレジスタ (CHKPNT_REG_i) に可能な状態

V	CP	GT	説 明
0	X	X	チェックポイントが割り付けられていない。チェックポイントが能動でない場合、CPは常にゼロにセットされる。
1	0	0	チェックポイントが割り付けられている。割り付け時にはCSN はこのチェックポイントのシリアル番号より小さく、CSN はまだチェックポイントを追いついて進んでいない。
1	0	1	チェックポイントが割り付けられている。割り付け時にはCSN がチェックポイントのシリアル番号より大きく、CSN はまだチェックポイントを追いついて進んでいない。
1	1	X	チェックポイントが割り付けられている。CSN はチェックポイントを追いついている。GTは有効でない（注意不要）。次のチェックポイントとも「11X」状態ならば、このチェックポイントをリタイヤさせることができる。

表3. 典型的なQビットの状態と遷移

Qの状態	状態の説明
000	チェックポイント割り付け中 $0 < \text{CSN} < 32$ ならば、現在形成されているチェックポイントの初期状態。
001	チェックポイントの割り付け中 $31 < \text{CSN} < 64$ 用に現在形成されているチェックポイントの初期状態。
010	割り付け以後、63/0境界を横切ってCSNが進んでしまったチェックポイントの状態。
011	割り付け以後CSNが63/0境界を横切って2回進んでしまったチェックポイントの状態。
1XX	CSNが63/0境界を横切って2回進んでしまい、CSNが現在32よりも小さいチェックポイントの状態。 Xは注意不要を示す。

B. チェックポイントの割り付け

命令SN用のチェックポイントを割り付ける時には、チェックポイント割り付けロジックユニット351が、チェックポイント「1」が有効でリタイヤされていないことを示すためにCHKPNT_REG iでVビットを1にセットし、CPビットをゼロにセットする信号をチェックポイントレジスタとコントロールユニット352へ送る。割り付けられるチェックポイントは、ISUに送られて割り付けに使用できる次のチェックポイント番号を知らせる信号と、チェックポイントの形成を可能にさせる信号を含みチェックポイントを作る命令のシリアル番号を識別するISUからのDO_CHKPNTとによってCAL351内で決められる。チェックポイント割り付けの時にCSNの位置を示すため、ビットQは表3に示すように書き込まれる。この情報はリングの周

囲（例えばロケーション63から逆進してロケーション0まで）からのCSNの遷移を検出するために用いられる。割り付けられたチェックポイントは、チェックポ

イントのシリアル番号を識別し駆動信号としても働く信号DO_CHKPNT によって決定される。更に、チェックポイントに対応するシリアル番号はチェックポイント割り付けロジックユニット351 によって計算されるが、1 だけ大きくされたシリアル番号 ($NSN = SN + 1$) だけがレジスタに記憶される。シリアル番号NSN は2 種の異なる条件で用いられる。第1 に、NSN はICRU301 へのデータバスで大きくされたチェックポイントのシリアル番号になる。第2 に、NSN はチェックポイントに後続する命令に故障命令があるか否かを決定するのに用いられる。SNではなくてNSN ($SN + 1$) をセーブすることによってロジックを節約し、チェックポイントの後のシリアル番号を識別する信号の発生を高速化するのである。この代りに、大きくされたシリアル番号信号を発生するために設けられているレジスタとロジックにSNそのものを記憶することができる。

次のようなチェックポイント割り付け法が典型的である。チェックポイント形成信号が主張される (DO_CHKPNT) と、V ビットフィールドが「1」にセットされ ($CHKPNT_REG_i [V_フィールド] = 1$)、CP ビットフィールドがクリアされる ($CHKPNT_REG_i [CP_フィールド] = 0$)。次にチェックポイントが形成された命令のシリアル番号がコード化された4 ビットのチェックポイントロケーションベクトルから引き出されてチェックポイントのシリアル番号フィールドに記憶される ($CHKPNT_REG_i [SN_フィールド]$)。例えばCPU51 の実施例では、「0000」はチェックポイントロケーションNISNを示し、「0001」はNISN+1を示し、「0011」はNISN+2を示し、「0111」はNISN+3を示し、「1111」はNISN+4を示す。

C. チェックポイントのリタイヤ

チェックポイントはチェックポイントクリア/リタイヤメントロジックユニット353 でクリアまたはリタイヤされる。チェックポイント303 はサイクル毎にISU200によって要請される数だけのチェックポイントを必要とする。現在のチェックポイント (n) と次のチェックポイント (n+1) を通るCSN に応じてレジスタ352 のV ビットフィールドをクリアすることによってチェックポイントがリタイヤされる。チェックポイント割り付けロジックユニット353 は、チェックポイ

ントレジスタ352 でリタイヤされるチェックポイントを指示するために用いられる OLDEST_ACTIVE_CHECKPOINT と呼ばれるポインタを維持する。チェックポイントをリタイヤさせる手続きは次の通りである。チェックポイント (CKPSN) に記憶されているシリアル番号とユニットされたシリアル番号 (CSN) ポインタ (例えば CKPSN_CSN) との差を計算することによってリタイヤロジック353 に「CSN_PASSED」と呼ばれる信号が発生する。CSN は ICRU301 に維持され、各サイクルでチェックポイントユニット303 へ送られる。CSN がある与えられたチェックポイント (n) と現在その次にあるチェックポイント (n+1) を追い越して進んだ時か、チェックポイントのシリアル番号を追い越したマシンバックアップがある場合だけ、チェックポイントをリタイヤさせることができる。

本発明の構造と方法の一実施例では、CSN をすべてのチェックポイントのシリアル番号と同時に比較するロジックを供給することにより、CSN がチェックポイントを追いついて進んでいることが決められる。比較の際には、CSN がチェックポイントのシリアル番号を追い越したか否かを決定するのに単に試験よりも大きいことだけでは不十分な場合があるので、Aリング312 の特性の周辺のラップも考慮する (モジュロリング長)。

D. マシンバックアップに関するチェックポイントの保守

次のマシンサイクルを開始するために KILL_VEC 信号を用意するようチェックポイントユニット303 に知らせる WILL_BACKUP 信号と BACKUP_CHECKPOINT 信号をバックトラップ305 から受信すると、KILL_VEC はコミットされている命令を殺す。殺される命令は、チェックポイントの命令のシリアル番号を指定する BACKUP_CHECKPOINT 信号と最後のコミテッド信号から引き出される。CSN とバックアップチェックポイントの間にあるすべての命令が殺される。

CPU51 の一種の実施例では、BACKUP_CHECKPOINT はバックトラップ305 内のバックアップユニット402 によって発生される信号で、状態を回復するためにどのチェックポイントをバックアップするか CPU51 に知らせる。ISN とバックアップチェックポイントの間にある命令のどれを殺すかを指示するため、命令キルベクトル信号 (KILL_VEC) もバックトラップによって発生される。例えば、命令キル

ベクトルは殺される命令を識別するホットビットを持つマルチホット信号とすることができる。この代りに他の命令型式を使用することもできる。

チェックポイント303は、バックアップ中にICRU301がISUを調整することができるように、チェックポイントのシリアル番号を識別するチェックポイントレジスタ情報をICRU301に供給する。チェックポイントユニット303は、未完浮動小数点例外またはデータ区切り点例外が検出された時(CHKPNT_SN_INC)、ISNを調整するためにICRUが使用するシリアル番号も供給する。

E. タイムアウトチェックポイントの強化

従来のチェックポイント技術では、命令のデコードされたアトリビュートに基づいてチェックポイントが作られる。連続するチェックポイント（チェックポイント境界のついた）の間の命令の数が制

限されないように、チェックポイントは或る種の命令のアトリビュートに対してだけ作られる。命令実行例外が起こった時、故障命令以前のCPU51の状態を回復するためにCPU51は多数のマシンサイクルを費さなければならないかも知れない。従って、故障命令に達する前にCPU51が命令を再実行するか元に戻すために費す時間は、チェックポイントの境界内にある命令ウィンドウのサイズと命令の数の関数になる。ここでの命令ウィンドウのサイズとは、CPU51内で任意の時間に目立つことを許される命令の最大数のことである。（例示したCPU51では、任意の時間に64個までの命令が目立つことができる。）

本発明の構造と方法に強化する場合、チェックポイントはデコードされた命令のアトリビュート（だけ）に基づいて作られるのではなく、むしろCPU51内のタイムアウト状態によって作られる。タイムアウト状態はチェックポイントを作らずに経過した命令発行サイクルの数に基づいている。例えば、最近発行された命令が従来チェックポイントを作られない型であればチェックポイントは形成されていない。タイムアウトチェックポイントは命令の最大数をチェックポイントの境界内に制限する。命令ウィンドウのサイズがチェックポイント境界内にある命令の最大数よりも大きい限り、例外が生じた場合に、プロセッサは従来の命令デコードに基づくチェックポイント技術よりも速くチェックポイントされた状態を

回復することができる。更に、本発明のチェックポイントの強化により、CPU51の状態回復の命令ウィンドウサイズへの依存性が除去される。

タイムアウトチェックポイントは従来のチェックポイントを設置する構造と方法でも使用することができる。更に、タイムアウトチェックポイントは、本発明のバックアップ及びバックステップ構造と方法と同様、従来のプロセッサバックアップ技術と共用すること

ができる。

タイムアウトカウンタ355は、そのマシンサイクル中に少なくとも1個の命令が発行されたことを示すINSTRA_ISSUED信号をISU200から受信して命令発行サイクルの数をカウントする。チェックポイント（命令デコードに基づくチェックポイントまたは以前のタイムアウトチェックポイント）を作ることなく経過した命令発行サイクルの数が所定のカウンタ数を越えると、タイムアウト状態が生じたのでチェックポイントを形成すべしというTIMEOUT_CHKPNT信号をISUに送る。この時ISU200はチェックポイントを形成すべしと示すDO_CHKPT信号を発生する。前述したように、発行された信号がチェックポイントを必要とする型であるとISU200が決定した時にもこの信号を発生させることができる。チェックポイントが実際に形成された場合、そしてこのような場合だけ、ISUからのDO_CHKPNT信号によってカウンタ356がセットされる。チェックポイント境界内に許される命令の最大数は、カウンタの最大命令発行率（マシンサイクル毎に発行される命令の最大数）とタイムアウト値によって決められる。

タイムアウトチェックポイント上の変化が有利であろう。例えば、タイムアウトチェックポイントと同一サイクルに正当に発行される命令デコードが正当である場合、命令デコードに基づくチェックポイントが優先的に作られてタイムアウトカウンタをリセットするように働く。更に、この時すべてのチェックポイントが割り付けられてタイムアウトチェックポイントが正当になった場合には、命令デコード型のチェックポイントのために厳密な状態を維持するために要するようなマシン停止を行わずに、所定の規則（例えば1サイクル延期）に従ってタイムアウトチェックポイントを遅らせることもできる。

F. 任意の命令境界での厳密状態の維持及び回復

本発明の方法によれば、任意の命令境界で厳密状態を維持することができ、その命令境界に状態を回復するために要する情報量を最小にすることもできる。ここで言う「命令境界」とは、命令が発行される直前（または直後）にある連続する命令の組の中の点に対応する。各命令が命令境界を作る。

従来の命令のチェックポイントではプロセッサ内のチェックポイント境界（即ち、チェックポイントされた命令でだけ）厳密状態を維持することができるが、従来のチェックポイント法は各命令で用いられ、そのように用いられた場合でもサイクル毎に複数命令を発行するマシンに対してはサイクル毎に同数のチェックポイントが必要であるため、CPU51 の性能を改善するよりも阻害するであろう。本発明では、サイクル毎に1個の命令だけがチェックポイントを要するように種々の命令の発行規則が設定され、バックステップ法と構造がより細かい命令レベルの回復を行う。命令のチェックポイントは理論上の命令順序によって不正に変更されるかも知れない状態のようなアーキテクチャ（マシン）状態を保護し後に記憶するために命令のチェックポイントが使用される。

チェックポイント境界に厳密状態を維持する従来のチェックポイント法が知られている。しかし、マシン状態を捕捉し後に回復する方法としてだけ従来のチェックポイント法を使用すると、理論上順序外の実行中、間隔の狭い命令境界に厳密状態を維持することができない。

G. チェックポイントされるデータ量を減らすよう所定命令に対してマシンを同期させる方法

チェックポイントは、命令型式とは無関係な固定サイズまたはチェックポイントが作られているオペランドである。例えば、200個

の状態値を変えることができる命令を持つマシン(CPU)では、命令自体がせいぜい8個の値を変更できる場合でも、形成された各チェックポイントは100個の状態値を記憶しなければならない。本発明の実施例では、特定の命令の各々によって変更される状態の量と典型的な処理において特定命令の各々が発生する統計上の頻度を考慮して、すべてのチェックポイントのチェックポイントされたデータ

の量を減らすように構造と方法が設定される。

この設計技術を用いると、最適なチェックポイントサイズを決めてマシンに設定することができる。第1に、各命令によって変更できる状態が決定される。第2に、ある代表的な名目上の処理作業に対して各特定命令が発生する統計的な頻度が決められる。各命令用の状態記憶要件が検討される。設計の目標は、稀にしか発行されず頻繁に発生する命令よりも比較的多量のチェックポイントを必要とする命令を識別することである。チェックポイントする命令を決めるにはリニアまたはノンリニアな最適化技術を用いることができ、チェックポイントされた状態を減らすために最大チェックポイント記憶に基づく非算術的な規則を用いることもできる。単純化した例を次に説明する。

チェックポイントされる状態が減少している単純化した例
命令の組の中のどれかの命令によって変更できる状態

状態1	状態2	状態3	状態4	状態5	状態6	状態7	状態8	状態9	状態10
-----	-----	-----	-----	-----	-----	-----	-----	-----	------

命令Aによって変更(mod)できる状態

mod	—	—	mod	—	—	—	—	—	—
-----	---	---	-----	---	---	---	---	---	---

命令Bによって変更(mod)できる状態

mod	—	—	—	—	—	—	—	—	—
-----	---	---	---	---	---	---	---	---	---

命令Cによって変更(mod)できる状態

—	mod	mod	—	mod	mod	mod	—	mod	mod
---	-----	-----	---	-----	-----	-----	---	-----	-----

命令Dによって変更(mod)できる状態

—	—	—	—	—	—	—	mod	—	—
---	---	---	---	---	---	---	-----	---	---

命令A、B、C及びDの各々がチェックポイントされる場合には、各チェックポイントは状態1～10のすべてを記憶しなければならない。しかし命令Cは稀にしか実行されないので稀に行われるこの命令の実行中マシンを同期させることに

よってCPU51のスローダウンが必要である場合には、命令Cのチェックポイントを行わないことによってチェックポイントされる状態を10状態から3状態、即ち、状態1と状態4と状態8に減らす。

マシンを同期させると、ペンディング命令がコミットしリタイヤされるまでマシンの同期を要する命令の発行が遅れ、次に同期を要する命令が順次速続して実行される。マシンが同期モードで作動している時には、状態への逆書き込みが行われる前に例外条件が識別されるので、厳密状態を維持するために命令のチェックポイントをする必要はない。

H. チェックポイントされるデータ量を減らすようにレジスタリネームマップをチェックポイントする方法

更に、誤予測分岐、故障、実行エラーその他の処理中の異状からマシン状態が回復できるようにチェックポイントが発生された時の全機状態が記憶されるため、従来のチェックポイントは多量の情報を含んでいる。従来のチェックポイント法では変化する状態だけを

変更する試みは行われていない。

例えば、64個の命令を並行して発行することができる従来のチェックポイント法をCPU51内の各命令に適用すると、各命令に対して1個、つまり64個までのチェックポイントを維持しなければならず、4発行のマシンでは、マシンサイクル毎に4個ものチェックポイントを作らなければならないことになる。従来のチェックポイントに記憶される状態の特定の型と量はCPU51のアーキテクチャと命令の組によって異なるが、従来の各チェックポイントは任意の命令によって変更できる各状態用の状態情報を含んでいる。即ち、命令が状態を変更し得るか否かに拘らず、従来のチェックポイントは各変更可能状態用の状態情報を含んでいる。従来のチェックポイント法では、チェックポイント毎に変化する状態だけではなく、マシン状態を回復するに必要なすべてのマシン状態が各チェックポイントに対して書き換えられる。

本発明の方法では、レジスタがリネームされリネームマップがレジスタリネームマップに記憶されている。レジスタのリネームについては他の箇所の説明する

。レジスタ名のリネームは処理に利用できるアーキテクチャレジスタの数を効果的に増加するのに役立つもっと多数の物理レジスタにアーキテクチャレジスタをマッピングする方法である。本発明の方法では、データ値そのものではなくてリネームされたレジスタをチェックポイントとすることによって、新しいチェックポイント法に必要な記憶の量を減らす。前述した通り、以前に発行されたすべての命令が実行されコミットされ、その資源がリタイヤされる(ISN=CSN=RRP)ように、厳密状態を維持するために多量のチェックポイントされたデータを必要とする或る種の所定の命令はマシン同期を起させる。同期型の命令故障が誤予測されると、状態は変更されず、命令の実行結果が状態を変更する

ことを許される前に、故障が処理されるかまたは代りに命令通路が追跡される。

マシン状態が素速く回復されるように、コントロールレジスタを変更できる非同期を要求するすべての命令はチェックポイントされる。命令によって変更されたすべての状態は、前述のようにチェックポイントされてCPU51内に局所的に記憶される。更に、コントロールレジスタを変更しない命令を実行する状態はCPU51をバックステップすることによって記憶できる。第26図は、タイムアウトチェックポイントを形成する他のチェックポイント法に適用できる任意のチェックポイント強化法を含む本発明のチェックポイント法の実施例の概略的フローチャートである。

V. 誤予測及び例外からの回復

前に言及したように、第5図の命令パイプラインでは、プログラムコントロールの誤予測と例外が時折生ずる。この時にはエラーを検出して誤予測または例外を生じさせた命令の直前の状態にCPU51を戻すように操作しなければならない。

A. 同時的な複数の分岐/ジャンプアンドリンク命令の評価のためのウォッチポイントによる誤予測の検出

命令を理論上で実行できるプロセッサに対しては、命令を解決する前に分岐方向(採用または非採用)または分岐アドレス(目標アドレスまたは分岐命令の次のアドレス)を予測することができる。後にこれらの予測が誤っていたと判明すると、CPU51は以前の状態に戻り正しい分岐ストリームで命令の実行を開始する

。従来のプロセッサでは、1サイクルでの評価のためにしばしば複数の分岐が用意される。しかし、従来のプロセッサはサイクル毎に1個だけの分岐予測を評価する。従って、評価のために用意はされたが選択されなかった分岐評価は遅延されなければならない。分岐評価が遅延す

るとCPU51の性能が低下し、分岐をできるだけ早い時期に評価することによって性能を著しく改良することが出来る。

本発明のウォッチポイントの構造と方法は従来技術よりもいくつかの点で有利である。第1に、複数の予測分岐またはジャンプアンドリンク命令を同時にモニタまたは監視することができる。第2に、複数の実行ユニットからのデータ拡大バスを同時にモニタまたは監視して予測分岐またはジャンプアンドリンク命令が待っている複数のデータまたは計算されたジャンプアンドリンク(JMPL)アドレスをつかむことができる。第3に、分岐またはジャンプアンドリンク命令の複数の誤予測信号を発生することができる。第4に、単一の共有記憶領域内に交代の分岐アドレスまたは予測ジャンプアンドリンクアドレスを記憶することができる。第5に、ウォッチポイントによって誤予測が検出された時に命令を取り出すための正しい分岐アドレスまたはジャンプアンドリンクアドレスを送ることができる。

第27図を参照すると、ウォッチポイント(WPT)304は発行された各チェックポイント済みのコントロール転送命令の実行及び予測状態のモニタに責任がある。従って、WPT304は、分岐、無効化分岐と、BrXcc、条件BrIcc、条件BrFcc、JMPL rt(1)(rd=0)用の条件情報とコントロールレジスタ(CCR及びFSRレジスタを除く)への読取り/書き込みを含むジャンプーリンク命令を含んでいるDO_WTC HPT信号と共にDO_CHKPT信号を受信する。条件コード情報は後に、命令順序が正しく予測されたかそれとも元に戻す必要があるかを決めるのにWPT304によって使用される。WPT304については本明細書の他の箇所で詳細に論議する。

ウォッチポイントユニット304は、DFB62からのデータ拡大バスによって到来する実行完了信号のモニタまたは監視と、例えば分岐

誤予測のような所定の事象が検出された時をCPU51 に知らせることに責任を持ち。ウォッチポイント304 内のウォッチポイントレジスタ491 は多目的である。これらのレジスタは分岐予測とJMPL予測をチェックし、本来の予測が正しくない時にバックアップ後に命令を参照するために正しい分岐/JMPLアドレスをBRB59 へ送る。これらのレジスタはデータ拡大分配バスにあって数個のユニット(FXU, FPU, LSU 及びFXAGU)からの結果をモニタするので「ウォッチポイント」レジスタと呼ばれる。16個のウォッチポイントレジスタがあり、各チェックポイントに対して1個となっている。

整数実行ユニット(FXU)と、整数及びアドレス発生実行ユニット(FXAGU)と浮動小数点実行ユニット(FPU)とを持つCPUに関連して例示したウォッチポイントユニット304 を説明する。この説明からして、設定する実行ユニットの数を大きくしたり、小さくしたり、変えたりすることができることは当業者には自明であろう。特に、例示したCPU51 の実行ユニットをマッチさせるために、FXU とFXAGU が追加されるであろう。

第27図はウォッチポイントユニット304 内にある主な機能ブロックの機能ブロック図である。BRB59 内のフェッチユニット(FETCH)は命令を取り出してISSUE200に与え、命令のデコードに基づいて分岐とJMPL'sを予測する。ISSUE200はウォッチポイント304 と FSR/CCRFRNへDO_CHKPNT 信号を送り、ウォッチポイント304 へDO_CHKPNT 信号を送る。FSR/CCRFRNは条件コードのリネーム(CC_リネーミング)に責任がある。DO_CHKPNT はチェックポイントの形成を開始するMAKE_CHKPNT 信号と、形成されるチェックポイント番号を識別するNEXT_CHKPNT 信号とを含む。DO_WTCHPT 信号は分岐またはJMPL信号が発行され分岐の型を識別したごとと、発行された命令中の条件フィールドと、条件コード(CC) タグと、発行された命令中の

次の交代のプログラムカウンタアドレス/予測ジャンプアンドリンク命令アドレスをウォッチポイント304 へ知らせる。FETCH はまた、DO_WTCHPT 信号内のWR_A NPC 信号という形で分岐/JMPLアドレスをウォッチポイント304 へ知らせる。分岐またはJMPL或いは他のコントロール転送命令(分岐/JMPL)が発行されると、

ISU200は分岐/JMPL命令発行済み信号(DO_WTCHPT)を送るが、これは信号JMPLを含む発行済みの各予測分岐またはJMPL用の信号である。WP_ANPCは目標RAMに書き込まれ、PCロジック106によって計算された目標FPCを示す。ANPCは予測分岐命令とJMPL命令の命令デコード中に決められる。JMPL命令に対しては、DO_WTCHPT信号内のウォッチポイントへ送られたWP_ANPCが予測目標FPCになる。予測分岐命令に対しては、DO_WTCHPT信号内のウォッチポイントへ送られたWP_ANPCが交代のFPC、つまり交代の分岐方向のFPCになる。誤予測が生ずると、FPC、APC及びNAPCを計算するバックアップの間、ウォッチポイント内のANPC出力ロジックからの出力されるRD_ANPCが使用される。リネーム600(FSR/CCRFRN606と呼ばれる)の条件コード(CC)部分はCCレジスタのリネーミングを管理し、命令発行後にウォッチポイント304へCCリネーミング信号を送る。CCリネーミング信号は次に説明するようにCC_TAG_RENAMED_Cと、CC_TAG_Cと、CC_DV_Cと、CC_DADA_Cを含んでいる。

実行ユニット(FXU601, FXAGU及びFPU600)の各々はデータ拡大バスによって条件コードタグ(CC_TAG_F)と条件コードデータ有効(CC_DV_F)と条件コードデータ(CC_DATA_F)をウォッチポイント304とFSR/CCRFRN606の両方へ送る。FXUからのタグとデータ有効とXCCデータ信号とICCデータ信号は、FXU_XICC_TAGS_Fと、FXU_XICC_DV_Fと、FXU_XCC_DATA_FとFXU_ICC_DATA_Fを含んでいる。FXAGUからの信号は、FXAGU_XICC_TAGS_Fと、FXUAG_XICC_DV_Fと、

FXAGU_XCC_DATA_FとFXAGU_ICC_DATA_Fを含んでいる。FPUからの信号は、FPU_FCC_TAGS_Fと、FPU_FCC_DV_Fと、FPU_FCC_DATA_Fを含んでいる。XCC、ICC及びFCC条件コードの例と書式は例えばSPARC-V9に説明してある。

第28図を参照すると、ウォッチポイント304は、実行ユニット(FXU, FPU及びFXAGU)からのCCデータ拡大バスから直接、またはFSR/CCRFRNから到来するリネームされたCC信号から条件コードをつかむCC__グラビングロジック1000と、分岐またはJMPL命令がデコードされる度毎にBRB59から到来する予測条件コードデータを含むウォッチポイント情報の記憶に責任を持つウォッチポイント記憶素子1002及び関連する読取り/書き込みコントロールロジックと、実行ユニットから到

来する（直接またはリネームユニットを経由して）計算された条件コードまたは実際の条件コードの比較を行ってこれらの条件コードを個々のウォッチポイント記憶素子1002に記憶されている予測条件コードと比較することに責任を持つ評価ロジック（Evalロジック）とを含む。目標RAMとその関連するコントロール比較ロジック1003は、次に詳細に説明するように分岐またはJMPL誤予測からの回復に参加する。CCグラッピングロジック1000と、Evalロジック1001とWP_素子10002は、CPU51内に割り付けできる16個のウォッチポイントの各々に対して1個、つまり16個の別々の構造を含んでいる。唯1個だけの目標RAM1003データ記憶構造が供給される。

16個のウォッチポイントの各々に対して数個のデータ素子が記憶される。各ウォッチポイントに対して記憶されるデータは、ウォッチポイント活性化時にセットされるウォッチポイント有効ビット（WP_VALID[i]=1）と、JMPL命令から分岐命令を区別する命令型式データ（WP_JMPL[i]=1はJMPLがウォッチポイントを必要として

いることを示し、WP_JMPL[i]=0は予測分岐用にウォッチポイントが形成されていることを示す）と、命令中の条件フィールド（WP_COND[i]）と、WP_XCC、WP_ICC及びWP_FCC用の記憶装置とを含む。一種の典型的な実施例では、これらのデータ素子はラッチに記憶されている。CCタグはCC_TAG_ARRAYに記憶され、「ANPC」（交代用の次のプログラムカウンタアドレス）／予測JMPL（ジャンプアンドリンク命令）アドレスは目標RAMに記憶されている。有効はウォッチポイントロジックによって書き込まれ、分岐の型、COND、CCタグ及びANPC/JMPLアドレスはISU200から受信するDO_WTCHPT信号から引き出される。DO_WTCHPTに応じて各分岐/JMPL命令に対してウォッチポイント記憶素子491が活性化され、フィールド選択書き込みロジックがDO_WTCHPT信号をデコードし、ウォッチポイントに分岐型式と、CONDフィールドとANPC/予測JMPLアドレスとを書き込む。

有効ビットを「1」にセットすることによってウォッチポイント項目（エントリ）を活性化させることができるが、すべてのチェックポイント素子とそのサイクルで既に割り付けされてしまっている場合にはウォッチポイント（及びチェッ

クポイントを必要とする命令の発行は延期される。複数マルチポイント素子を同時に能動(有効)とすることができ、有効な (VALIDフィールドセット) すべてウォッチポイント項目は各サイクルで同時に評価することができ、誤予測信号を発生することができる。データ記憶に割り付けされるチップ領域を減らすため、ウォッチポイント304 は分岐命令用のANPCの個かJMPL命令用の予測ターゲットアドレスどちらかを同一の記憶位置に記憶することができる。割り付けられた多目的ウォッチポイントレジスタの実際の使われ方は、ISU200からのDO_WTCHPT 信号の内容、即ち、ウォッチポイント項目が予測分岐用として作られて

いるかまたは予測JMPL命令用として作られているかによる。ウォッチポイントレジスタは自ら割り付けを解いて分岐となり、ジャンプ予測が評価する。

条件分岐命令の場合のように命令によって誤予測が生ずる可能性のある時には、ウォッチポイントユニット304 によって必ずウォッチポイント素子が割り付けされる。各ウォッチポイント素子是对应するチェックポイント素子を持っているが、逆は必ずしも真ではない。例えば条件分岐のように命令によって誤予測が生ずる可能性がある場合には必ずウォッチポイント素子が割り付けされるので、チェックポイント是对应する能動ウォッチポイント項目を持たなくてもよい。チェックポイントは誤予測を生ずる可能性のある命令 (「SAVE命令のような命令または周期的な「タイムアウト」チェックポイントが分岐命令以外の命令用に作られた時) 以外の命令に対しても割り付けできるので、すべてのチェックポイントがウォッチポイントを持っている訳ではない。ウォッチポイントはウォッチポイントユニット304 内の複数のウォッチポイント素子432 のうちの1個に記憶される。チェックポイント情報を記憶するために1個のチェックポイントレジスタが設けられる。しかし、実際のチェックポイント済みマシン状態は前述のようにCPU51の全体に亘って局所的に記憶される。表4は一組の典型的な命令用としてシリアル番号によってチェックポイント中に記憶された情報の型と対応するチェックポイントの例を示す。

条件コードグラビングロジックユニット492 は、DFB62 内の実行ユニットからのデータ拡大バスに現われる時の条件コード (CC) を捕捉することと、命令が発

行された時に予測され現在ウォッチポイント素子491に記憶されている条件コードと捕捉された条件コードを比較する評価ロジックユニット493へルートを指定することに責任

を持つ。捕捉された条件コードが予測条件コードと合う場合は、命令が正しく予測されたので元に戻される必要はない。しかし、グラビングユニットによって捕捉された条件コードが予測条件コードに合わない場合には、命令が誤予測されたので元へ戻さなければならない。

表4. ウォッチポイントとチェックポイント記憶の単純化した例

No.	チェックポイント	ウォッチポイント
0	SN=33, BRZ、マシン状態	BRZ, CCレジスタがCCレジスタタグを待つ。
1	SN=41, ADD、マシン状態	—ウォッチポイント不要
2	SN=44, JMPL、マシン状態	JMPL、レジスタタグ (jmpl アドレス計算の結果)
3	SN=47, SAVE、マシン状態	—ウォッチポイント不要
4	—フリーチェックポイント	—フリーチェックポイント
----		----
14	SN=28, FBGE、マシン状態	FBGE, FCCがレジスタタグを待っていた。
15	--フリーチェックポイント	--フリーチェックポイント

実行ユニットからの条件コードの捕捉と比較は下記のものを含む数種のファクタによって複雑になる。即ち、命令順序から外れる実行、各実行ユニット内の命令実行待ち行列、異なる命令型式の実行に要するマシンサイクルが変数であること、同一の命令型式でも実行資源には実行データを待つ必要に関する変動性である。上記のファクタとその他のファクタは一般に、命令の結果と状態が実際にデー

タ拡大バスにいつ現れるかに関して不確実性を生む。従って、本発明は命令が発行された時に同一のマシンサイクル中または任意の後のサイクル中にバスに現れる可能性のある条件コードデータを捕捉する構造及び方法を含む。また、本発明はリネームされたレジスタ用の条件コードを捕捉する構造及び方法を提供する。レジスタ依存性を除くため、例示したCPU51では条件コードのリネーミングを含めたレジスタのリネーミングが行われる。従って、ウォッチポイントユニット304はCCデータとタグを直接データ拡大バスから受信し、その後CCRFRN内でCCレジスタが更新されてから、FSR/CCRFRN606からCCデータとタグを受信する。

第27図ではDFB62に内に2個の固定小数点実行ユニット(FXUとFXAGU)と1個の浮動小数点実行ユニット(FPU)が示されているが、更に多数または少数を設置することができ、これらは任意の型であってよい。各実行ユニットはCCデーター拡大バスによってCCRFRN606とウォッチポイント304に接続される。CCデーター拡大バスは単位実行ユニット(FXU, FXAGU 及びFPU)によって計算されたばかりのCCデータの結果を搬送する。CCデータはCCRFRNを更新するために使われる。ウォッチポイント304はCCデーター拡大バスをモニタまたは監視し、後のサイクルでCCRFRNからCCデータを貰うのを待つのではなく、バスから直接にCCデータをつかむ。このようにバスから直接にCCデータをつかみ取ることによって分岐評価が速くなる。

ウォッチポイント304は分岐誤予測を検出すると、PSU300内のPLSM307へ誤予測信号WP_MISPRED_VECを送る。PSU300は前述したようにCPU51の主要状態を追跡制御する。PSUがウォッチポイントから誤予測信号WP_MISPRED_VECを貰うと、PSU300内のPLSM307が誤予測と実行例外のうちで一番初期のものを選択し、本明細書の他の箇所

に説明するように逆追跡手続きを開始する。

第25図はウォッチポイント304及びISBと、SFR/CCRFRNと、DFB, BRB内の実行ユニット(FXU, FXAGU及びFPU)と、PSU内のPLSMへのインターフェースのより詳細な機能ブロック図である。これらの主要な機能ブロックを次に説明する。

1. ウォッチポイント要素の活動化

ウォッチポイント要素及び制御論理ユニット (WP-要素) 1002は、WP活動化／非活動化制御論理2012及び16の要素の各々がXCC, ICC, FCC, COND, JMPL／分岐及び妥当性インジエータに関する情報を記憶している16のウォッチポイント要素セット2013を含んでいる。分岐／JMPLが発行され予測された時点で、予測済み分岐、取消し分岐及びジャンプーリンク命令についての条件コード情報を内含するISU200によってDO_WTCHPT信号がアサートされる。「分岐を予測する」というのは、「分岐の方向（とられたか又はとられていないか）を予測する」ことを意味し、「JMPLを予測する」というのは、JMPLがつねにプログラムのフローを変更する（つねにとられる）ことから、「JMPL標的アドレスを予測する」ことを意味する。分岐／JMPL命令を発行する場合、典型的 CPU51は、チェックポイント番号により識別されたチェックポイントを作成する。ひとたび作成されると、チェックポイントは CPU51が1つのチェックポイント場所と次のチェックポイント場所の間に存在する命令のための全ての命令実行を完了してしまうまで、活動状態にとどまる。プロセッサは一度に多数の活動中のチェックポイントをもつことができる。チェックポイントの形成は発行された分岐／JMPL命令に制限されず、例えば1つのトラップをとるときといったようなその他のケースで形成される可能性もある。

各々のウォッチポイント要素2012は、1つのチェックポイントエントリに対応する。典型的なCPU51は16のチェックポイントエントリを有し、従ってウォッチポイント304 は16のウォッチポイント要素をもつ。ウォッチポイント活動化／非活動化論理 (WADL) 2012は、ISU200からのDO_CHKPNNT 及びDO_WTCHPNNT信号に応じてウォッチポイント要素を活動化させることを担当している。DO_CHKPNNT は、チェックポイント形成を開始させるMAKE_CHKPNNT 信号及び、形成されるべきチェックポイント番号を識別するNEXT_CHKPNNT 信号を含んでいる。WADL2012は同様に、予測済み分岐／JMPL命令が解決された時点でウォッチポイント要素を非活動化することを担当している。

ISU2000が分岐命令を発行すると、ISU200は、チェックポイントが形成されるべきであるということを表示するMAKE_CHKPNNT 信号をアサートし、1つのチェッ

クポイント番号がPSU300によって割当てられNEXT_CHKPNT により指定される。図50に示されているように、1つのウォッチポイント要素を活動化するためには、NEXT_CHKPNT がデコーダ434 によって復号され、ANDゲート435 内でJSU200からのDO_WTCHPT から誘導されたDO_PREDICT信号との論理積がとられる。この出力はDO_PREDICT_VEC[i]と呼ばれる。信号DO_PREDICT_VEC[i]はWP_VALID[i]をセットする。なおここで「[i]」は16のウォッチポイント番号のうちの1つを表示する。図50は同様に、ウォッチポイント出力論理2102の一部分を含む、WP_ACTIVE_VEC及びWP_MISPRED_VEC論理を結びつけられたウォッチポイントユニットの特定の実施形態についての付加的な構造的詳細をも示している。

ISU200は、分岐/JMPL命令が発行された時点でウォッチポイントに対して複数の信号を送る。これらの信号は、XCCに左右される分岐として、ICCに左右される分岐として、FCCに左右される分岐として、ジャンプーリンク命令(JMPL)として、その命令を識別し、その分岐について条件コードがファイルされる(COND)。これらの

信号はウォッチポイント304 により受信され、DO_CHKPNT の成分信号としてNEXT_CHKPNT によって指定されたウォッチポイント要素の中に記憶される。ウォッチポイント要素「i」がウォッチポイント活動化制御論理(WACL) 2012によって割当てられると、これらの信号はそれぞれWP_XCC[i], WP_ICC[i], WP_FCC[i], WP_JMPL[i] 及びWP_COND[i]レジスタの中に保たれる。各々のウォッチポイント要素は、それが非活動化され次にもう1つの分岐/JMPL命令によって再度活動化されるまで、これらのレジスタ値を保持する。

典型的な CPU51は、各サイクル毎に複数のチェックポイントを作成することができ、又1サイクルにつき複数の分岐/JMPL命令を発行することができる。従って、複数のウォッチポイント要素を活動化させることができる。例えば、CPU51が同じサイクル内で2つのチェックポイントを作成し2つの分岐を発行できる場合、DO_WTCHPT, XCC, ICC, FCC, JMPL, COND, DO_CHKPNT信号が2セット又はそれと同等の情報内容が必要とされる。

条件コード書式及びフィールド定義及びJMPL(ジャンプ&リンク) 命令の書式

及び定義は、本明細書中ですでに記録されている通り、SPARCV9アーキテクチャマニュアル内に提供されている。浮動小数点条件コードはビット0及びビット1を含む2ビット書式をもつ、整数条件コードは、xccについてビット7-4（それぞれN, Z, V, C）に、又 ICCについてはビット3-0（それぞれN, Z, V, C）として提供されている。分岐が発行され「とられた」ものと予測された時点で、COND値は分岐命令の条件フィールドデータと同じである。分岐が送信され「とられていない」と予測された時点で、CONDフィールドデータのビット-3が逆転され、残りのビットは変更されず、これらはCOND値となる。

図28は、CC-捕捉論理1000, Eval（評価）論理1001, WO-要素記

憶及び制御ユニット1002及び標的 ram1003を示す。ウォッチポイント304 の内部構造のより詳しい表示を示す。図29は、評価準備完了 (EVAL_READY) 及び評価条件コード (EVAL_CC) 論理を含むCC-セレクト論理1006、アレイ晚期一致論理1004、現行一致論理1005を含む、ウォッチポイント304 の異なる概略的部分を示す。

図30は、分岐予測及び評価のタイミング図を示す。ウォッチポイントは、各々予測された分岐/JMPL情報を保持し予測の正しさを評価することのできる複数のウォッチポイント要素を有する。サイクル1では、分岐が発行される。サイクル2では、複数のウォッチポイント要素の1つ（要素「i」と仮定する）が活動化される。その後、WP_ACTIVE[i]がアサートされる。サイクル5では、FXUユニットは、その分岐を左右しウォッチポイント要素が待っているCCデータを復帰させている。サイクル5では、ウォッチポイント要素iはCCデータを捕捉する。サイクル6では、ウォッチポイント要素がCCデータを捕捉していることからWP_ACTIVE_VEC[i]はアサート解除され、EVAL_READY[i]がアサートされる。EVAL_READY[i]がCC評価を有効化する。予測が正しい場合には、EVAL_TRUE[i]がアサートされる。そうでない場合、EVAL_READY[i]はアサートされない。この例では、タイミングダイアグラムは、予測が正しくないことをアサートする。次にWP_MISPRED_VECがアサートされ、PSU300内のPLSMへ送られる。PLSMは全てのウォッチポイント要素（典型的実施形態では16の要素）からのWP_MISPRED_VEC[i]信号及び実行ユニットからの実行トラップ信号の中から優先順位をとり、最も早期の事象を決定

する。サイクル7では、PSU300は、CPU51を以前の状態までバックアップさせるDO_BACKUP 信号をアサートする。DO_BACKUP 信号は、CPUにバックアップを実施するよう知らせるMAKE_BACKUP 及び、そこまでバックアップが作成されるべきチェックポイント番号

を識別するBACKUP_CHKENT 信号を含んでいる。図29は、複数のウォッチポイント要素のうちの1つに付随するウォッチポイントユニットの特定の実施形態についての付加的な構造的詳細を示している。

2. データ順方向送りバスから直接のCCデータのウォッチポイント捕捉

CC-捕捉（グラビング）論理1000は、データ順方向送りバスから条件コードデータ有効及び条件コードタグを受理しこれらを現行のマシンサイクル中に ISB から受理した条件コードタグと一致させようとするCCタグ現行一致論理2001を含んでいる。典型的な論理回路が図31にCCタグ現行一致論理の一実施形態について示されている。

FSR/CCRFRN606 は、以下の信号をウォッチポイントに送る：CC_RENAMED, CC_TAG_C, CC_TAG_RENAMED_C, CC_DV_C及びCC_DATA。CC_RENAMEDは、CCが現行のマシンサイクル内で発行された命令によって修正された場合にアサートされる。CC_TAG_Cは、リネーム済みCC物理タグを識別するものの、現行のマシンサイクル内で発生するCC修正は除外される。CC_TAG_RENAMED_Cも同様に、リネーム済みCC物理タグを意味するが、現行サイクル中に発生するCC修正は内含される。CC_DV_Cは、CC_DATA が、先行サイクルによって修正されたCCデータを有するものの現行サイクル内で行なわれたCC修正を除外する場合に、アサートされる。従って、CC_VD_C=1である場合、先行サイクルによって行なわれた全てのCC修正はCC_DATA 内に反映される。その上、CC_DV_C=0である場合、CC_DATA 値は最新のものではなく、FSR/CCRFRNは、最新のCCデータを捕捉する前に先行するサイクルによって発行された命令の完了を持っている。FSR/CCRFRN606 によってウォッチポイントに送られる信号は、表6に記されている。

表 6 FSR/CCRFRN606は以下の信号をウォッチポイントに送る

信号	内容説明
CC_RENAMED	現行マシンサイクル内で発行された命令によりCCが修正された時点で表明される。条件コードセレクト論理によって使用される。
CC_TAG_C	リネーム済CC-物理タグを識別するが、現行のマシンサイクル内で発生するCC修正を除外する。現行の一致について使用される。
CC_TAG_RENAMED_C	リネーム済CC物理タグを識別するが現行サイクル内で発生するCC修正を内含する。アレイ内に書込まれ、アレイ一致のために使用される。
CC_DV_C	CC_DATA が先行サイクルによって修正されたCCデータを有する場合に表明されるが、現行サイクル内で行なわれたCC修正を除外する。条件コードセレクト論理により用いられる。
CC_DATA	<p>CC_DV_C = 1 である場合、先行サイクルによって行なわれた全てのCC修正は、CC_DATA [7 : 0] 内に反映される。</p> <p>CC_DV_C = 0 である場合、CC_DATA 値は最新のものではなく、FSR/CCRFRNは、最新CCデータを捕捉する前に先行サイクルによって発行された命令の完了を待っている。条件コードセレクト論理によって使用される。</p>

図31では、比較回路2006、2008及び2010は各々、リネーム済みcc物理タグを識別するCC_TAG_C信号を受理するが、現行のマシンサイクル中に発生するcc修正は

除外する。比較回路の各々は同様に、データ順方向送りバスから直接条件コードタグを受理する。すなわち FPUからFPU_FCC_TAG_Fを、FXUから FXU_XICC_TAG_Fをそして FXAGUから FXAGU_XICC_TAG_Fを受理し、ここで FXAGU_XICC_TAG_F及び FXU_XICC_TAG_F は両方共整数条件コードの XCC及び ICCの両方の部分を内含する。比較の出力は ANDゲート2007, 2009及び2011に送られ、制御信号として役立つデータ有効信号(FPU_FCC_DV_F, FXU_XICC_DV_F又は FXAGU_XICC_DV_F)との論理積がとられる。データ有効信号がアサートされたならば、そのとき、比較回路によって決定された一致は有効であり、CC現行一致信号(FPU_CURR_MATCH, FXU_CURR_MATCH 又は FXAGU_CURR_MATCH)がアサートされる。

図31に示されているように、CCリネーム済みタグアレイ一致論理2002は、各々の実行ユニット(FPU, FXU, FXAGU)について AND論理機能を実行するための論理回路2016及び比較回路2015 16セットから成る1つのセットを含んでいる。現行一致論理2001はデータ順方向送りバスからの信号をCC_TAG_Cと比較する一方、アレイ一致論理ユニット2002は、リネーム済みCC物理タグを識別するものの現行サイクル中に発生するCC修正を含むCC_TAG_RENAMED_Cとデータ順方向送りバスからの信号の各々を同時に比較する。CC_TAG_RENAMED_Cは、DO_PREDICT_VECと現行マシンスサイクル内で発行された命令によりCCが修正された時点でアサートされているCC_RENAMEDとの論理積をとることにより、生成された書込み有効化信号の制御下で、16のCCタグアレイ2003記憶要素のうちの1つの中へ書き込まれる。サイクル毎に、CC_TAG_RENAMED_Cの記憶された値は、何らかの一致を識別する目的で、FPUからのFPU_FCC_TAG_F, FXUからのFXU_XICC_TAG_F及び FXAGUからのFXAGU_XICC_TAG_Fと比較される。比較回路2016の出力は、制御信号として役立つデータ有効信号(FPU_FCC_DV_F, FXU_XICC_DV_F、又は FXAGU_XICC_DV_F)と論理積がとられる。データ有効信号がアサートされると、そのとき比較回路によって決定された一致は有効であり、CCアレイ一致信号(FPU_ARRAY_MATCH[i], FXU_ARRAY_MATCH[i]及び FXAGU_ARRAY_MATCH[i])がアサートされる。従って、CC一捕捉論理は、複数の未解決分岐評価を同時に監視することができる。

図31は、CC捕捉論理を示す。CC_TAG_ARRAYは16のレジスタを含み、各レジスタは16のチェックポイントのうちの1つに対応し、CC_TAG_Cビット数と同じラッチ数をもつ（ここで我々は4ビットラッチを仮定している。1つの分岐を発行し予測する場合、CC_TAG_RENAMED_Cは、DO_WTCHPT から誘導されたDO_PREDICT_VEC[i]により指定されるCC_TAG_ARRAYの場所内に書き込まれる。CC_RENAMEDは、AND論理ゲートといった書込み有効化及びアドレスセクタ論理2004内で書込み有効化信号として役立つ。

この記述においては、プロセッサは2つの固定小数点実行ユニットすなわちFXU及びFXAGUそして1つの浮動小数点実行ユニットFPUをもつものとして仮定されている。典型的な実施形態においては、FXUのみがJMPL命令を実行することが仮定されている。各々のユニットはCCデータ順方向送りバスを有する。FXU CCデータ順方向送りバスは、FXU_CC_TAG_VALID、FXU_CC_TAG及びFXU_CC_DATAを含む。FXUが、CCを修正するようなその実行を終わろうとしているとき、FXU_CC_TAG_VALID がアサートされ、FXU_CC_TAGはそのリネームされたCC物理タグを有し、FXU_CC_DATAはCCの結果データを有する。同じことはFXAGU及びFPUにもあてはまる。FSR/CCRFRNユニット内の条件コード(CC)データは次のサイクル内でこれらのCC

データ順方向送りバスを用いて更新される。ウォッチポイントは同様に、ウォッチポイント要素がCCデータを待機している場合にデータ順方向送りバスからのCCデータを捕捉する。CC_TAG_CはFXU_CCTAGと比較され、その一致信号はFXU_CC_TAG_VALIDとの論理和がとられ、出力信号はFXU_CC_CURR_MATCHと呼ばれる。同様にCC_TAG_ARRAY内のレジスタの内容は、FXU_CC_TAGと比較され、その一致信号は、FXU_CC_TAG_VALIDとの論理積がとられ、出力信号はFXU_CC_ARRAY_MATCH[i]と呼ばれる。FXAGU及びFPUについては、FXUと同じ一致機能が存在する。ここで3つの命令発行シーケンス例を基準にして、捕捉論理のオペレーションについて記述する。

CC捕捉論理1000は同様に、そのウォッチポイント要素内の分岐を左右し、アサートされた時点でCCデータ信号（例えばCC_DATA[7:4]）を選択し同様に次

のサイクル内での評価のためにそのデータ信号をラッチングする条件コードに基づいて信号を生成することをも担当するccセレクト論理2005（各ウォッチポイントに1つずつの16層）をも含んでいる。条件コードセレクト論理2005の1実施形態のための典型的論理回路は、図32の中に提供されている。このccセレクト論理2005のオペレーションについては、以下の3つの例に関連して記述されている。

評価論理1001は、図25に示されている通り、評価準備完了論理（EVAL_READY）2100、評価真論理（EVAL_TRUE）2101 及びウォッチポイント出力論理（WP出力論理）2102を内含する。図33は、ccセレクト論理2005からの1組のセレクト信号（例えば、SEL_BR_XCC[i]）及び AND論理ゲート2106の出力からの有効化信号（EVAL_ENABLE）を受理する EVAL_READY論理2100に付随する付加的な構造的詳細を示している。EVAL_ENABLE は、WP_ACTIVE_VEC[i], WP_JMPL, FXAGU_JMPL 及び FXAGU_CHKPTに基づいてアサートされる。EVAL_READY論理

2100は、評価が準備完了していることを表わす i 番目のウォッチポイントのための EVAL_READY[i]信号を出力する。

図34は、CC_EVAL 論理2105を含む EVAL_TRUE 論理2101に付随する付加的な論理的詳細を示す。EVAL_TRUE 論理は、分岐命名又はJMPL命令が適正に予測されたか否かを決定することを担当する。CC_EVAL2105 は、ccセレクト論理2005からの EVAL_CC[i]及び、ISBから受理されウォッチポイント要素2013の中に記憶された WP_COND[i], WP_XCC[i], WP_ICC[i]及び WP_FCC[i]を受理し、SPARC-V9マニュアル内に記述された規則を用いた比較に基づいて2つの信号を評価する。これらが一致する場合、分岐真信号（BR_TRUE[i]）がその i 番目のウォッチポイントについて生成される。EVAL_TRUE 論理2101は、同様にJMPL一致論理2201による JMPL_MATCH出力（図35参照）を、ISBから受理されウォッチポイント要素2013内に記憶された WP_JMPL[i]と比較する。JMPL_MATCHが WP_JMPL[i]と一致する場合、JMPLは適正に予測されており、JMPL_TRUE[i]がアサートされる。BR_TRUE[i]と JMPL_TRUE[i]の論理和がとられ、EVAL_TRUE[i]信号を生成する。単一の命令のみについて1つのウォッチポイントが形成され、従って、BR_TRUE[i]又は JMPL_TRUE[i]の1つのみがアサートされることになる。

WP出力論理2102が図50に示されている。PSU300からのEVAL_READY[i], EVAL_TRUE[j], WP_VALID[i]及びKILL_VEC[i]は、WP出力論理に入力され、これがWP_ACTIVE_VEC[i]及びWP_MISPRED_VEC[i]信号を出力する。

図35は、TARGET_RAM及びJMPL評価論理に付随する付加的な構造的詳細を示している。JMPL評価論理は、rd=0とした復帰タイプのJMPL命令を評価した。ANPC出力論理2103は、正しい計算値である FXU_DATA の値を、本書で記述されている優先順位決定スキーマ及び記

憶されたWP_ANPC の比較に基づいて PSUPLSMによって誤予測(MISPRED)がアサートされているか否かに基づき BRBフェッチユニットまで送るため、RD_ANPC を選択する。

3. ウォッチポイントオペレーション例

a. ウォッチポイントオペレーションー例1：

表7は、サイクルX、X+1、及びX+2の中で実行された、例1のための命令のリストである。サイクルXでは、4つの命令が発行され、そのうちの1つ(addcc)がCCを修正する。このとき、リネームされた-cc物理タグ#7が割当てられる。サイクルX+1では、ここでも4つの命令が発行され、そのうちのいずれもCCを修正せず、addccが実行されてサイクルX+1内でCCデータを復帰させる。サイクルX+2では addccによって修正されたCCデータはFSR/CCRFRN内にある。サイクルX+2では4つの命令が発行され、そのうちのいずれも分岐前にCCを修正せず、その後、分岐命令(br, xcc)が発行される。この場合、サイクルX+2では、現行サイクル内でCCを修正する命令が全く発行されないことからCC_RENAMED=0である。その上、最も晩期のCCタグが#7でありこれがサイクルXで割当てられることから、CC_TAG_C=CC_TAG_RENAMED_C=7である。最後に、最も晩期のCCがFSR/CCRFRN内にあることから、CC_DV_C=1である。

1つの分岐を発行しているとき、FSR/CCRFRN内で分岐を左右するCC_DATA が有効である。従って、CC_DV_C は分岐の発行と同じサイクルでアサートされる。分岐が xccにより左右されると仮定してみよう。そのとき SEL_BR_XCC(i) がアサートされ(図32参照)、信号はCC_TDATA[7:4]を選択し、データは次のサイ

クルにおいてEVAL_CC[i]内でラッチされる（信号CC_DATA[7:4]はXCC_DATA_C及びICC_DATA_Cであるか又はFCC_DATA_Cである）。又EVAL_READY[i]は次のサイクルにおいてアサートされる（図33参照）。CC評価は、準備完了状態となる。

b. ウォッチポイントユニットオペレーションー例2:

表8は、サイクルX及びX+1内で実行された例2についての命令リストである。サイクルXでは、4つの命令が発行され、そのうちの1つ(addcc)がCCを修正する。その後、リネーム済みCC物理タグ#7が割当てられる。サイクルX+1では、ここでも4つの命令が発行され、そのうちのいずれも分岐前にCCを修正せず、そのうちの1つが分岐命令である。この場合、サイクルX+1では、現行サイクルでCCを修正するような命令は全く発行されていないため、CC_RENAMED=0である。又、最も晩期のCCタグが#7であり、それが先行サイクルで割当てられているため、CC_TAG_C=CC_TAG_RENAMED_C=7である。ただし、addccによって修正される最も晩期のCCデータはFSR/CCRFRNにおいてCCレジスタ内に書込まれないことからCC_DV_C=0である。

（サイクルX+1において）1つの分岐を発行する場合、サイクルXで発行されている命令「addcc」はCCを修正し結果データはFSR/CCRFRNにおいてCCレジスタ内に書込まれないことから、FSR/CCRFRN内でその分岐を左右するCC_DATAは有効でない（CC_VD_C=0）。我々はFXU実行ユニットによりサイクルX+1で「addcc」が実行されつつあり、FXUがサイクルX+1でそのデータ順方向送り母線上でCCデータを復帰させている、と仮定している。その後、サイクルX+1でFXU_CC_CURR_MATCH(図31参照)がアサートされる。サイクルX+2でSEL_FXU_XCC[i]（図32参照）がアサートされ（図32参照）、信号はFXU_CC_DATAを選択し、データはEVAL_CC[i]内でラッチされる。又、サイクルX+2でEVAL_READY[i]がアサートされる（図33を参照）。次にCC評価が準備完了状態となる。

FXU_CC_CURR_MATCH比較のためにはCC_TAG_RENAMED_CではなくCC_TAG_Cが使用されることに留意されたい。分岐が発行されるのと同じサイクル内で、CCを修正

するような命令が発行される場合、CC_TAG_Cは最も晩期のCC物理タグではない。しかし、CC_RENAMEDがアサートされ信号は FXU_CC_CURR_MATCHを抑止する（図32参照）。このことはすなわち、分岐が発行されるのと同じサイクルにてCCを修正する命令が全く無い場合には、FXU_CC_CURR_MATCHを使用することができる、ということを意味している。FXU_CC_TAG_CURR_MATCHのためにCC_TAG_C[3:0]を用いることは、CC_TAG_CがCC_TAG_RENAMED_Cよりも早い信号であることから、経路遅延を減少させる一助となる。

c. ウォッチポイントユニットオペレーション—例3：

表9は、サイクルX及びX+1で実行される例3についての命令リストである。サイクルXでは、4つの命令が発行され、(addcc)の1つがCCを修正する。このときリネーム済み—CC物理タグ#7が割当られる。サイクルX+1では、ここでも4つの命令が発行され、そのうちの1つ(subcc)が分岐前にCCを修正し、それらのうちの1つは分岐命令である。リネームされたCC物理タグ#8は命令subccに割当てられる。この場合、サイクルX+1では、現行サイクル内でCCを修正する命令が発行されているため、CC_RENAMED=1である。先行サイクルによる最も晩期のCCタグが#7であることからCC_TAG_C=7であり、現行サイクルでCCを修正する命令が発行されておりリネームされたCC物理タグ#8が割当てられることから、CC_TAG_RENAMED_C=8である。同様に、subccによって修正されている最も晩期のCCデータがFSR/CCRFRN内でCCレジスタ内に書込まれていることから、CC_DV_C=0である。

(サイクルX+1において) 1つの分岐を発行する場合、サイク

ルXで発行されている命令「addcc」はCCを修正し結果データはFSR/CCRFRNにおいてCCレジスタ内に書込まれないことから、FSR/CCRFRN内でその分岐を左右するCC_DATAは有効でない(CC_DV_C=0)。その上、命令「subcc」はサイクルX+1で発行され、その後CC_RENAMEDがアサートされる。我々は、FXU実行ユニットによりサイクルX+2で「subcc」が実行されつつあり、FXUがサイクルX+2でそのデータ順方向送り母線上でCCデータを復帰させている、と仮定している。その後、サイクルX+2でFXU_CC_ARRAY_MATCH[i]（図31参照）がアサートさ

れる。サイクルX+3でSEL_FXU_XCC[i] (図32参照) がアサートされ (図32参照)、信号はFXU_CC_DATAを選択し、データはEVAL_CC[i]内でラッチされる。又サイクルX+3でEVAL_READY[i]がアサートされる (図33を参照)。次にCC評価が準備完了状態となる。この例では16のウォッチポイント要素が存在し、各々の要素は、いずれかのサイクルにおいて各分岐評価について並行なccを捕捉することができる。

表7 例1についてのFSR/CCRFRNからの命令及び信号

サイクル	命令	FSR/CCRFRN信号
X	move mul addcc <#7 div	
X+1	move move move (#7 実行済み) move	
X+2	mul move move br, xcc	CC_RENAMED=0 CC_TAG_C=7 CC_TAG_RENAMED_C=7 CC_DV_C=1

表8 例2についてのFSR/CCRFRNからの命令及び信号

サイクル	命令	FSR/CCRFRN信号
X	move mul addcc <#7 div	
X+1	mul (#7 実行済み) move move br, xcc	CC_RENAMED=0 CC_TAG_C=7 CC_TAG_RENAMED_C=7 CC_DV_C=0

表9 例3 についてのFSR/CCRFRNからの命令及び信号

サイクル	命令	FSR/CCRFRN信号
X	move mul addcc <#7 div	
X+1	mul move subcc <#8 br, xcc	CC_RENAMED=1 CC_TAG_C=7 CC_TAG_RENAMED_C=8 CC_DV_C=0

4. 分岐命令の評価

分岐が発行されてから1サイクル後に、WP_VALID[i]がアサートされる。WP_VALID[i]=1でありEVAL_READY[i]=0そしてKILL_VEC[i]=0である場合、WP_ACTIVE_VEC[i]がアサートされる(図29参照)。「WP_ACTIVE[i]=1」は、チェックポイントiを用いる分岐がすでに発行されているもののまだ解決されていないことを意味する。値が0となったならば、それは分岐が解決された(完了した)ことを意味する。WP_ACTIVE_VECはICRU301に送られ、どの分岐/JMPL命令が完了されるかを決定するために用いられる。

分岐評価が準備完了状態になった時点で、EVAL_READY[i]がアサートされEVAL_CC[i]は、捕捉されたcc値である。このとき、EVAL_CC[i]及びWP_COND[i]はEVAL_TRUE LOGIC2101内でCC_EVAL論理中へフィードされ(図34参照)、分岐は評価される。分岐を発行するときに作成された予測が正しい場合、BR_TRUE[i]がアサートされる。そうでない場合、これはアサートされない。1つの分岐を評価するとき、分岐タイプを知るために、WP_XCC[i], WP_ICC[i], WP_FCC[i]信号も同様に使用される。BR_TRUE[i]はJMPL_TRUE[i](以下で説明する)と論理和がとられ、EVAL_TRUE[i]が生成される。WP_VALID[i]=1でEVAL_READY[i]=1でEVAL_TRUE[i]=0でKILL_VEC[i]=0である場合、WP_MISPRED_VEC[i]はアサートされる(図29参照)。「WP_MISPRED_VEC[i]=1」というのは、以前に作成された分岐予測がまちがっていることを意味する。PSUはこの信号及び実行エラー信号を受理し、最も早期の誤予測又は実行エラー条件を選択し、次にBACKUP信号をアサートして以前の状態までバックアップする。

PSUは、バックアップが発生しバックアップ時点の後にとられたチェックポイントをキルする必要がある場合に、KILL_VEC[i]信号

をアサートする。KILL_VEC[i]がアサートされた時点で、反応するウォッチポイント要素も同様にキルされる。「キルされる」というのは、WP_ACTIVE_VEC [i] 及びWP_MISPRED_VEC[i]が抑制されることを意味する。WP_ACTIVE_VEC 及びWP_MISPRED_VECは、WP_MISPRED_VEC [0-16] を同じベクトル内でコード化できるように、マルチホットであり得る。このことはすなわち、一度に最高16のウォッチポイント要素（この例では）が活動中であり得又一度に誤予測を検出できるということをしている。

分岐を発行し予測するとき、FETCHがWR_ANPC を通してウォッチポイントに、分岐標的アドレス又は分岐の次のアドレスを送る。予測が「とられた」場合、WR_ANPC が、分岐の次のアドレス（とられていないアドレス）である。予測が「とられていない」場合、WR_ANPC は分岐標的アドレス（とられたアドレス）である。予測が誤っていることが判明した場合、ウォッチポイントは、適正な分岐経路から命令を再度フェッチするためRD_ANPC を介してアドレスをFETCHまで送り戻す。WR_ANPC 値はTARGET_RAM内に記憶される（図35参照）。TARGET_RAMは16のエントリをもち、各々のエントリは各々のチェックポイントに対応し、1つのWR_ANPC アドレスを記憶することができる。この RAMの書込み有効化信号及び書込みアドレスは、MAKE_CHKPNT 及びNEXT_CHKPNT である。MAKE_CHKPNT は、チェックポイントを作成するときにアサートされる。BACKUP信号がアサートされた時点で、TARGET_RAMの読取りアドレスのためにBACKUP_CHKPNT が用いられる。BACKUP_CHKPNT は、CPU51がそれまでバックアップするチェックポイントを指定する。又TARGET_RAMの読取りデータがRD_ANPC を介して FETCHまで送られる。

5. JMPL-LINK(JMPL)命令の評価

JMPL命令を発行し予測するとき、FETCHは、ウォッチポイントに

対しWR_ANPC を介して予測されたJMPL標的アドレスを送る。ウォッチポイントはアドレスを保ち、実行ユニットにより計算上の正しいアドレスとこのアドレスを

比較することによって適正さを評価する。予測が誤っていたことが判明した場合、ウォッチポイントは、正しいアドレスから命令を再度フェッチするためRD_ANP Cを介してFETCHに計算上の正しいアドレスを送る。予測されたJMPL標的アドレスは分岐の場合と同じ方法でTARGET_RAM内に記憶される。

我々は、FXAGU実行ユニットのみがJMPL標的アドレスを計算するものと想定している。FXAGUがJMPL命令の実行を終えた時点で、FXAGU_JMPLがアサートされ、FXAGU_CHKPNTはJMPLのチェックポイント番号を指定し、FXAGU_DATAは、計算上の正しいJMPL標的アドレスを内含する。このとき、FXAGU_CHKPNTによって指標付けされたTARGE_RAM内の予測されたJMPLアドレスが読みとられ、読取られたデータはFXAGU_DATAと比較されることになる。比較結果はラッチされ、ラッチ済み信号はJMPL_MATCHである(図35)。予測上のアドレスと計算上のアドレスが同じである場合、JMPL_MATCHがアサートされる。このとき、JMPL_MATCHはWP_JMPL[i]との論理値がとられ、出力はEVAL_TRUE[i]となる(図34)。FXAGU_JMPLがアサートされてから1サイクル後に、EVAL_READY[i]もアサートされる(図33)。

分岐命令の場合と同じ要領で、JMPLについてWP_ACTIVE_VEC[i]及びWP_MISPRED_VEC[i]が生成される。WP_MISPRED_VEC[i]がアサートされてから1サイクル後に、PSU300は、ウォッチポイント304に対する2つの信号すなわちBACKUP_CHKPNT及びMISPRED(ただしバックアップがJMPLの誤予測のせいである場合のみと同数のBACKUPを送る。MISPREDがアサートされた場合、ウォッチポイントは、FETCHに対し正しいJMPLアドレスを含むラッチされたFXAGU_DATA信

号を2回送る(図35参照)。TARGET_RAMは次の2つの目的のために使用される；その1つは分岐の代替的地址を記憶するためそしてもう1つは予測されたJMPL標的アドレスを記憶するためである。これら2つの用途は排他的であり、従って、この実施のためには1つのTARGET_RAMだけで充分である。

発明力ある構造及び方法は、(1)複数の予測済み分岐又はジャンプ&リンク命令のウォッチング(監視)を同時に開始するための構造及び方法；(2)実行ユニットからのデータ順方向送り母線を同時にウォッチングすることにより、予測済み分岐又はジャンプ&リンク命令が待っている複数の条件コードデータ又は

計算上のジャンプ&リンクアドレスを捕捉するための構造及び方法；(3) 単独又は(2)と組合わせた形で、分岐又はジャンプ&リンク命令の複数の誤予測信号を同時に生成できるようになるための構造及び方法；(4) 単独又は(2)と組合わせた形で、上記2つのケースで共有される1つの記憶装置内に代替分岐アドレス又は予測されたジャンプ&リンクアドレスを記憶するための構造及び方法；(5) 単独又は(4)と組合わせた形で、誤予測が発生した場合に命令フェッチのため正しい分岐アドレス又はジャンプ&リンクアドレスを送るための構造及び方法；そして(6) ccデータを捕捉するためのタグ比較を2つの部分すなわちデータ順方向送り母線タグと比較した先行サイクル内のccタグ及びデータ順方向送り母線タグと比較した現行発行ウインドウ内の現行リネーム済みccタグという2つの部分に分割することによって、クリティカルパスをスピードアップするための構造及び方法(単独又は(2)と組合わせた形)、を提供する。後者の場合、リネーム済みccタグのラッチされた信号を比較のために用いることができ、従ってこれはタイミングクリティカルではない。図36は、複数の同時未解決分岐評価のための発明力あるウォ

ッチポイント方法の一実施形態の概略的流れ図である。

B. 例外検出

例外は、発行又は実行中に発生する可能性がある。かくして例外には発行トラップと実行トラップが含まれる。以下では、発行トラップと実行トラップの検出について記述する。

1. 発行トラップの検出

ISU200は同様に、命令の発行を実施する発行トラップを検出する。これらの発行トラップは、同期化又は非同期化発行トラップであり得る。当業者であれば、同期化及び非同期化とされる発行トラップのタイプは、さまざまな基準に基づいて選択され得、設計者の選択の対象となりうるということを認識することだろう。

例えば、非同期化発行トラップは、標準的には、往々にして発生し、かつ前述の要領での CPU51のそれに対する同期化がプロセッサの速度を低下させることに

なるような発行トラップである。従って、このような種類のトラップは、機械的に同期化されエントリを受けることはない。これらの発行トラップとしては、SPARC-V9アーキテクチャマニュアル中に記述されているfill_normal, fill_other, spill_normal, spill_other, clean, window, trap_instruction (つねに (T A%go tsimm7) 命令の中間にあるトラップ)、unimplemented_ldd, unimplemented_std 及び illegal_instructionが含まれていると考えられる。

非同期化発行トラップは、標準的に、往々にして発生せず、かつそれについて CPU51を同期化することによってプロセッサの速度が著しく影響を受けないような発行トラップである。これには、SPARC-V9 instruction_access_error, instruction_access_exception, privileged_opcode, privileged_action, fp_exception_other, fp_disabled, emulation_trap、及びtrap_instruction (条件付きトラップ (TCC) 命令) 発行トラップといったような発行トラップが含まれると考えられる。

これらの発行トラップの或るタイプのものが発行段で発生したか否かを決定するために、ISU200は制御レジスタ (CNTRL-REG) フィールドを制御レジスタファイル800 から受理する。図17に示されているように、制御レジスタファイル800 は、クリーンウインドウ (CLEANWIN) レジスタ801、回復可能ウインドウ (CANRESTORE) レジスタ802、セーブ可能ウインドウ (CANSAVE) レジスタ803、ウインドウ状態 (WSTATE) レジスタ804 及びプロセッサ状態 (PSTATE) レジスタ805 を含む SPARC-V9特権的レジスタを収納している。これは又、TICKレジスタ806 及び浮動小数点レジスタ状況 (FPRS) レジスタ807 を含むSPARC-V9非特権的レジスタをも収納している。CNTRL_REGフィールドは、SPARC-V9アーキテクチャマニュアル内に記述されCLEANWIN, CANRESTORE, CANSAVE, WSTATE, TIC、及びFPRSレジスタ801~807 により提供されているCLEANWIN, CANRESTORE, CANSAVE, WSTATE_NORMAL, WSTATE_OTHER, PSTATE_PEF, PSTATE_PRIV, TICK_NPT、及びFPRS_FEFフィールドを含んでいる。

ここで図7に戻ると、ISU200は、FR_INSTRs_BRPs 命令を復号し、SPARC-V9発行トラップのいずれかが発生したか否かを決定するべくSPARC-V アーキテクチャマ

ニュアルに従って CNTRL_REGフィールドを利用する。最も早期の発行ウインドウスロット内で命令によりひき起こされた発行トラップのみが取られることになる。従って、単数又は複数の発行トラップが検出された時点で、最も早期のスロット内にある発行トラップ誘発命令のスロットに先立つ発行ウインドウスロット内の一部の命令のみがISU200によって発行され得る。発行トラップ誘発命令及び発行トラップ誘発命令のスロットの後の一部のFR_INST_BRP_0-3 命令は、(trap_instruction) 発行トラップ

をひき起こしたTA又はTcc 命令が発行されるということを除いて、発行されない。

発行トラップが起こった時点で、ISU200は、トラップが発生したことを表わす ISSUE_TRAP 信号をPSU300に対して出力し、前述の発行トラップのうちのいずれが発生したかを識別する。ISU200により非同期化発行トラップが検出された場合、その検出時点で直ちにISU200により ISSUE_TRAP 信号が出力されることになる。しかしながら、同期化発行トラップがISU200により検出された場合、発行トラップ誘発命令がスロット0内にくるまでISU200によってISSUE_TRAP 信号が出力されることはなく、CPU51は前述の要領で同期化される。非同期化発行トラップをひき起こす Tcc命令の場合、Tcc命令は、CPU51が同期化された時点で初めて発行され得る。

2. 実行トラップの検出

図8を参照すると、発行済み命令の実行中に、FPU600、FXU601及びLSU603によって検出されるエラーが発生する可能性がある。前述した通り、FPU600、FXU601及びLSU603は、命令の実行中にエラーが発生したか否かを表示する ERR_STAT 信号を ISB61のPSU200に出力する。LSU及び FXUからの実行エラーは、PSU300により容易にとり扱うことができる。しかし、次に記述するように、浮動小数点例外は特殊な取り扱いを必要とする。

図8を参照すると、前述の通り、FPU600は、資源利用可能性に基づいて予測上の及び／又は実際のPC順序外で浮動点命令をout-of-order実行する。浮動小数点命令の実行中、FPU600は実行トラップ（すなわちエラー）を検出することができ

る。しかしながら、FPU600は予測上の及び／又は実際のPC順序外で命令をout-of-order実行し得ることから、それらが引き起こす実行トラップも又予測上及び／又は実際のPC順序外であり得る。浮動小数点実行トラップをあたか

もそれらが実際のPC順に発生したかのごとく適切に取扱い検出するために、PSU300は、図37に示されているように浮動小数点例外(FPEXCEP)リング又はユニット306を含んでいる。

図38に示されている通り、FPEXCEPリング306は64の記憶要素又はエントリを伴うデータ記憶構造900を含んでいる。各々の記憶エントリは64の命令通し番号のうちの1つに対応し、命令識別(FP)フィールド、浮動小数点トラップタイプ(FTT)フィールド及び現行浮動小数点例外タイプ(CEXC)フィールドを有する。図37に示されている通り、データ記憶構造はFFFPフィールドを記憶するための64のアドレス可能な記憶要素をもつFPフィールドリング、FTTフィールドを記憶するため64のアドレス可能な記憶要素を有するFTTフィールドリングそしてCEXCフィールドを記憶するための64のアドレス可能な記憶要素を有するCEXCフィールドリングを形成する。FP、FTT及びCEXCフィールドリングのアドレス可能な記憶要素の各々は命令通し番号に対応する。

FPビットは、対応する命令がSPARC-V9浮動小数点タイプの命令であるか否かを表示するべくセット(「1」)又はクリア(「0」)される。FPEXCEPリング306内の浮動小数点命令としてFPビットにより識別された命令については、FTTフィールドは、発生したSPARC-V9浮動小数点トラップのタイプを識別する。SPARC-V9アーキテクチャマニュアル内に表示されている通り、これにはいかなるトラップも含まれず、IEEE_754例外、未実施_FPOP、未終了_FPOP、シーケンスエラー、ハードウェアエラー、及び無効_fpレジスタエラーが含まれている。当業者であればわかるように、CPU51はこれらのトラップタイプのいくつかを特定、検出及び取扱いすることなく実施でき、その場合、FPEXCEPリング306のFTTフィールドは、所要記憶空間が少なくすむように、FPEXCEPリング306のFTTフィールドをより小さなものにすることができる。その上、IEEE_754__

例外が発生したことを FTTフィールドが表示している浮動小数点命令については、CEXCフィールドは、IEEE標準754-1985及びSPARC-V9アーキテクチャマニュアルに従ってまさに発生したIEEE_754_例外の単数又は複数のタイプを識別する。かくして、CEXCフィールドは、1つのオペランドが不適正であるか否かを示すためのビット(nvc)、オーバーフローが発生したか否かを表わすためのビット(ofc)、アンダーフローが発生したか否かを表わすためのビット(ufc)、ゼロ除算が発生したか否かを表わすためのビット(dzc)、及び不正確な結果が発生したか否かを表わすためのビット(nxc)を含んでいる。しかしながら、浮動小数点トラップタイプのその他のタイプ又は浮動小数点トラップ無しのタイプを結果としてもたらず浮動小数点命令については、CEXCフィールドは、IEEE_754_例外のいずれのタイプも発生しなかったことを表示する。

前述し図11にも示されているA-リング312及びM-リング324と同様に、FP EXCEPリング306は、図37に示されているように「モジュロFP EXCEP-リングレジスタ長」算術演算（ここではモジュロ64算術演算）を用いてデータ構造に沿ってポインタが移動させられる循環リング又はラップアラウンドデータ構造として実施される。当業者であれば、循環データ構造が有利であるものの、これは、本発明にとって不可欠なことではなく、本発明の特徴を実施するその他のデータ構造を用いることも可能であるということがわかるだろう。

ここで再び図38を参照すると、各々の発行段の間、ISU200は、フェッチされたFR_INST, BRP_0-3 命令のどれが発行されたかを示す前述のISSUE_VALID 信号を出力する。同時にこれは、発行された命令のうちのどれが浮動小数点命令であるかを識別するEP_ISSUE信号

を出力する。これらの信号は、FP書込み(WR)論理901及びFTT書込み(WR)論理902により受理される。さらに、FPWR論理901及びFTTWR論理902はICRU301から、前述のNISNポインタを受理する。

FPWR論理901及びFTTWR902はNISNポインタ及びISSUED_VALID信号を用いて、ISU200により発行されたばかりの命令のためのFTTフィールド及びFRビットに対応するFP EXCEPリング内の場所をアドレスする。浮動少数点命令であるものとして

FP_ISSUEによって識別された命令の各々について、FRWR論理901は、それが浮動小数点命令であることを表わすため対応するFPビットをセットする。又、浮動小数点命令でないものとしてFP_ISSUEによって識別された発行済み命令については、FPWR論理901は、それが浮動小数点命令でないことを表示するため対応するFPビットをクリアする。

図8を参照すると、前述の通り、FPU600、FXU601、FXAGU602及びLSU603は発行済み命令を実行し、実行が完了した時点でERR_STAT信号を出力する。図39に示されているように、ERR_STAT信号は、FPU600によって実行され完了した浮動小数点命令についての通し番号、チェックポイント番号及びエラー状況フィールドを含むFP_ERR_STAT信号を含む。エラー状況フィールドは、実行済みの浮動小数点命令の結果をもたらされる浮動小数点トラップタイプ又は1つももたらされない場合には浮動小数点トラップ無しを、そして浮動小数点命令により引き起こされた現行の浮動小数点例外又は例も引き起こされなかった場合には現行浮動小数点命令無しを表示する。

その上、図39を参照すると、FSR/CCRFRN606は、SPARC-V9浮動小数点状況レジスタ(FSR)を含む。FSRは、上述のIEEE_754例外の各々について1つのマスキングビットを含むトラップ有効化マスク(TEM)フィールドを含み、これらの例外のうちの単数又は複

数のものがマスキングされうようになっている。TEMフィールドはFPU602に提供される。IEEE_754_例外が起こると、FR_ERR_STAT信号は、この例外がマスキングされるべきタイプのものであることをTEMフィールドが表示した場合には、1つの例外が発生したことをPLSM307に対して表示しない。しかしながら、FP_ERR_STAT信号はそれでもFPEXCEPユニットに対して、このようなIEEE_754_例外が発生した(すなわちFTT)ことと同時にそれがIEEE_754_例外のどのタイプであるか(すなわちCEXC)を表示する。

完了済み浮動小数点の各々について、FTTWR論理902は、FP_ERR_STAT信号により識別されたFTTデータをFPEXCEPリング306の対応するFTTフィールド内に書込むため、FP_ERR_STAT信号内に提供された通し番号を用いる。その上、CEXC

書込み (WR) 論理903 は、EP_ERR_STAT 信号によって識別されたCEXCデータを FPEXCEPリング306 の対応するCEXCフィールド内に書込むために、この通し番号を用いる。

読取り及び累積例外計算 (RD and AEXC compute) 論理904 は、PSU300の ICRU301から RRPポインタ及びCOMMIT_VALID信号を受理する。RRPポインタは、前のマシンサイクル内で退去させられた最後の命令をポイントし、COMMIT_VALID信号は RRPポインタが前のマシンサイクル内をどれほど遠くまで前進するか (すなわち退去させられた命令の数) を表示する。

各々のマシンサイクル中、RD及びAEXC計算論理904 は RRPポインタ及びCOMMIT_VALID信号を用いて、最後のマシンサイクル内で退去させられた各命令について FPEXCEPリング306 内のFPビット及び FTT及びCEXCフィールドを読みとる。その後、浮動小数点命令 (FPビットにより識別されたような) である退去済み命令について、RD及びAEXC計算論理904 は、そのCEXCフィールドを論理和してAEXCフィ

ールドを計算する。要約したとおり、マスキングされていないIEEE __754 __例外をひき起こす浮動小数点命令は退去されておらずPLSM307による実行トラップ順序決定を結果としてもたらすことになることから、AEXCフィールドは、最後のマシンサイクル内で退去させられた浮動小数点命令の全てによって累積されたマスキング済みIEEE __754 __例外を識別する。

さらに、RD及びAEXC計算論理は、読取りFPビットから、現行のマシンサイクルの中で退去させられた命令のうちのいずれが、退去されるべき最も晩期の浮動小数点命令であるかを決定する。現行のマシンサイクル内で単数又は複数の浮動小数点命令が退去させられた場合、RD及びAEXC計算論理904 は、計算上のAEXCフィールドを含む信号及び図39内の FSRのfsr.aexrフィールド内に書き込むべきであることを表示する信号を含むWR_CEXC_AEXC_FTT信号を出力する。その上、WR_CEXC_AEXC_FTT信号は、同様に、ほとんどの現行の退去済み浮動小数点命令のためのFTT 及びCEXCデータを含む信号及び図39内の FSRのfsrftt及びfsr.cexcフィールドの中にこのデータを書き込むべきであることを表示する信号をも含んでいる。この状況下で、FTTデータは、いくつかの理由で浮動小数点トラップ無しタイプ

を表示する。まず第1に、実行例外をひき起こさない浮動小数点命令は、退去される可能性があり、又浮動小数点トラップ無しタイプを表示する FPEXCEPリング内の対応する FTTフィールドを有することになる。第2に、IEEE_754 例外トラップを表示する FTTフィールドを有するもののFSR607の TEMフィールドによってマスキングされた単数又は複数の対応するIEEE_754 例外をひき起こした浮動小数点命令は、これらの実行例外のいずれかが発生したことについて PLSM307が警告を受けていないことから、退去される可能性がある。

前述のとおり、PLSM307は同様にFR_ERR_STAT 信号も受理する。これらの信号が、例外トラップが発生したことを表示した時点で、PLSM307により実行トラップ配列決定プロセスが開始され、こうして CPU51はトラップ誘発命令に先立つ命令において同期化されることになる（すなわち RRP=CSN=ISN）。

CPU51が同期化された時点で、PLSM307は FPEXCEP_CHP信号を生成する。FPEXCEP_CHK信号に応じて、RD及びAEXC計算論理902 は、RRPポインタを用いて、RRP+1にある命令のための FPEXCEPリング306 内の FP, FTT、及びCEXCフィールドを読みとる。

読取りFTT フィールドが、浮動小数点トラップ無しを表示した場合、RRP+1にある命令は、浮動小数点例外をひき起こす原因ではなかったことになる。この命令では、いくつかの他の種類の例外が発生した可能性もあることから、RD及びAEXC論理904 は、fsr.ftt, fsr.cexc 及びfsr.aexcフィールド内に例も書き込まれるべきではないことを表示する信号を内含するように、WR_CEXC_AEXC_FTT信号を出力する。

しかしながら、読取り FTTフィールドが浮動小数点トラップ発生を表示した場合、RRP+1での命令が、浮動小数点例外の原因であったことになる。このとき、RD及びAEXC論理904 は、RRP+1での命令のための FPEXCEPリング306 から読みとられた FTTフィールドを含む信号及び図39内のFSR607の fsr.fttフィールド内に読みとられるべきであることを表示する信号を含むように、WR_CEXC_AEXC_FTT信号を出力する。かくして、浮動小数点命令によってひき起こされたトラップタイプは、浮動小数点実行トラップ取扱いルーチンがトラップを適切に取扱うべ

く記憶FSR (STFSR) 命令で FSRにアクセスできるような形で、FSRの fsr.fttフィールド内に含まれている。

さらに、読取り FTTフィールドが IEE_754_例外が発生したことを表示した場合、RD及びAERC論理904 によって出力されたWR_CEXC_AEXC_FTT信号は、RRP+1での命令のための FPEXCEPTリング306 から読みとられたCEXCフィールドを含む信号及び図39内のFSR607の fsr.cexcフィールド内に読みとられるべきであることを表示する信号を含む。その結果、浮動小数点命令によって引き起こされた現行の例外タイプは、ここでトラップ取扱いルーチンが記憶FSR (STFSR) でFSR607のこのフィールドにアクセスしトラップを適切に取扱うことができるような形で、FSRの fsr.cexcフィールド内に含み入れられる。

図37を参照すると、この場合、浮動小数点命令は予測された及び／又は実際のPC順外でFPU600によりout-of-order実行されることから、RRP+1における命令は、実行トラップ順序決定プロセスを PLSM307に開始させた障害発生命令ではなかった可能性がある。換言すると、浮動小数点命令は、より早期に発行された（すなわちより早期の通し番号をもつ）もののより晩期に実行されたもう1つの浮動小数点命令によって引き起こされた実行トラップよりも早く検出された実行トラップを結果としてもたらした可能性がある。しかしながら、より晩期に発行されたもののより早期に実行された命令は、より早期の命令が1つの例外を結果としてもたらした場合にマシンサイクルに先立って順序決定しながら PLSM307による実行トラップ順序決定の開始をひき起こし、その後 PLSM307はその処理へと切り替わることにな。かくして、実行トラップ順序決定はそれでも、あたかもその実行トラップが実際のPC順で発生したかのごとく、より晩期に検出された実行トラップがより早期に検出された実行トラップに代って実際に取扱われるという結果をもたらす。

FPEXCEPTユニットについてSPARC-V9アーキテクチャという情況下

で記述してきたが、当業者であれば、浮動小数点命令の機械的実行が関与するようなあらゆるアーキテクチャのためにこれを実施することができる、ということ

も理解できるだろう。

C. プロセッサをより早期の状態にバックトラッキングすることによる回復

ウォッチポイント304 が誤予測命令を検出した時点、又は DFB62内で実行例外が発生し、PSU300内の実行例外(etrapp)をトリガーした時点で、CPU「バックトラック」が開始される。プロセッサをバックトラッキングすることには、以下でより詳細に説明する通り、プロセッサバックアップ及び／又はプロセッサバックステップが含まれる可能性がある。1つのマシンサイクル中、1つ以上の誤予測及び実行例外が発生し得る。典型的な CPU51においては、全ての機械的に実行された制御転送命令（例えば、予測された分岐命令）がチェックポイントされる。このような誤予測された命令はつねに「チェックポイント境界」に存在する。チェックポイント済みの誤予測命令からの回復は、発行時点における誤予測命令に対応するチェックポイントの中に記憶されたマシン状態を回復することによって達成できる。実行例外は、誤予測とは異なり、全ての命令がチェックポイントされるわけではないことから、チェックポイント済み命令に対応しないかもしれない。一般に、実行例外を生成する命令が、あらゆる「命令境界」に存在し得る。1つのチェックポイント境界は、機械的に発行された制御転送命令についてののみ1つの命令境界に対応する。機械的に発行された命令の全てがチェックポイント形成を要求するわけではなく、むしろPC不連続性を作り出すような機械的制御転送命令だけである（ただし、制御転送命令がマシン「同期化」命令として選択され、本書に記述のとおり選択的にチェックポイントされる場合は除く）ということをお願いしたい。

従って CPU51がバックトラックされる要領は、回復を開始した条件及び誤予測又は例外がチェックポイントされた命令に対応しているか否かによって左右される。

本書で使用される「バックアップ」という語は、チェックポイントの中に以前に記憶されたマシン状態を回復することによってチェックポイント境界においてマシン状態を回復し、次に CPU51命令の発行及び実行を再開するべく適切な行動をとるための構造及び方法のことを言う。本書で使用されているように、バック

アップには、チェックポイントされた命令から順方向の命令のその後のin-order再発行及び再実行が関与していてもいなくてもよい。ここで使用される「バックステップ」という語は、可能な場合にはバックアップと組合わせてあらゆる命令境界でマシン状態を回復し次に CPU51命令発行及び実行を再開するべく適切な行動をとるための構造及び方法のことを言う。

実行トラップ (etraps) に遭遇した場合、マシンは、その精確な状態を保存するため障害発生命令までバックトラックする。チェックポイント境界上に存在する誤予測と異なり、etrapsは、障害発生命令に達するべくバックステップ、バックアップ又はその両方の組合わせを必要とする。オペレーションの実行の必要性以外にはバックステップ又はバックアップオペレーションの順序付け又は頻度にはいかなる制約条件もない。組合わせて用いられた場合、マシンのバックアップ及びバックステップは、該予測及び実行例外からの効率の良い回復方法を提供し、しかも或る種の状況下では、有利にも、例外をひき起こす命令又はチェックポイント済み命令と障害発生命令の間の命令の再実行を強制することなく、これを行なうことができる。

プロセッサのフローの中に変化があった場合 (例えば、Bcc, FBcc, jump1及び同様の命令) にはつねに、チェックポイントが割振られる。典型的な CPU51は、割当てのために最高16のチェックポイントが利用可能な状態にある。バックアップはマシン状態をチェックポイント境界へ回復するために行なわれる。あらゆる命令境界にマシン状態を回復するためにバックステップが行なわれる。典型的なマシンは、利用可能なハードウェアによりリサイクルあたり最高4つの命令をバックステップすることに制限される。バックアップはマシン状態を回復するための粗いが迅速なバックトラッキング機構を提供し、バックステップは、細かい命令境界においてマシン状態を回復するための機構を提供する。マシンは、バックアップ又はバックステップのあらゆる組合わせを通して、より早期の状態までバックトラックすることができる。

発明力あるバックアップ手順と合わせて発明力あるバックステップ手順を用いて精確な状態回復を実現するためには、2つの規則を遵守しなければならない。

まず第1に、命令ストリーム内で、アーキテクチャ制御レジスタを修正し得る命令に遭遇した場合、CPU51が、命令の実行に先出し同期化されなくてはならないか、又はマシンがその命令をチェックポイントしなくてはならない。この第1の規則が遵守された場合、バックステッピングは、いかなるアーキテクチャ制御レジスタの修正も更新も決して要求しない。第2に、プログラムカウンタの不連続性を作り出す可能性のある全ての命令は、チェックポイントされなくてはならない。この第2の規則が遵守された場合、バックステッピングには、PCをバックステップ数だけ減分し、より早期の対応するプログラムカウンタで割当てられるにつれてマシン資源（RRF302内に維持された資源を含む）を回復することが関与する。

PSU300内に機械的に位置設定されたバックトラックユニット（BKTR）305が、適正な CPU51の機能及び回復に必要とされるようなマシンバックアップ及びバックステップを実施することを担当する。図40に示されている通り、BKTR305は、マシンバックアップ、単数又は複数のマシンバックステップ又はバックアップとバックステップの組合わせのいずれか誤予測及び例外からの CPU51の回復にとって適しているかを決定することを担当するバックトラック制御ユニット（BTCN）401を含んでいる。BTCN401は、DFB62内の各々との実行ユニット及びウォッチポイント304からSN、エラー及び状況情報を受理する。バックアップユニット（BKUP）402は、マシンバックアップを開始し順序決定することを担当し、バックステップユニット（BKST）403はマシンバックステップを開始し順序決定することを担当する。CPU51は、命令発行を停止し、例外又は誤予測をひき起こす命令に適する通りバックアップ及びバックステップを用いてバックトラックしなくてはならない。バックステップを用いて CPU61をバックアップするため及びマシンをバックステップするための構造及び方法について以下でさらに詳しく記述する。

1. チェックポイント境界へのプロセッサのバックアップ

マシンバックアップは、（1）PSU300内のPLSM307がRED Modeへのエントリを試みたとき、又は（2）ウォッチポイント304が誤予測

命令 (mispredicted instruction) を検出したときに起動される。また、(3) DFB 62 から障害命令 (faulting instruction) (実行例外 (execution exception)、即ち e-トラップが検出されたときは、マシンのバックアップが起動される場合もあり、そうでない場合もある。バックアップ手続きは、バックアップのために実行されたチェックポイント命令 (checkpointed instruction) の選択を含めて、バックアップを起動する理由によって異なる。実行例外の原因となる障害命令がチェックポイ

ント境界に一致しない場合には、バックステップがバックアップに続いて行われる。

各マシンサイクル毎に、バックアップユニット (BKUP) 402 は、データフォワードバスに関する命令の実行状態を示す情報 ERR_STAT を DFB 62 から受信すると共に、ウォッチポイント 304 からは誤予測命令に関する情報を受信する。これらの信号は、命令のシリアル番号 (SN)、必要に応じてマシンのバックアップを命ずるチェックポイント、エラー発生の有無に関する指示、及びエラー形式に関する指示 (例えば、据置 (deferred)、データ中断、浮動小数点動作等) を識別する。FXAGU が乗算又は除算処理を行わない場合には、その実行ユニットからの ERR_STAT 信号は、一般にエラー情報は与えず、ERR_STAT 信号中の命令のシリアル番号 SN 及びチェックポイント番号に関する情報を与える。チェックポイントは、発せられる全ての命令に対して形成されるものではないが、各命令にはその発生時点で、バックアップチェックポイントに割り当てられる。このチェックポイントは、命令発生以前に PSU 300 内の ICRU 301 によって割り当てられ、命令発生時には ISU 200 によって命令と組み合わせられ、ISU 200 によって演算コード (opcode) 及び命令のシリアル番号と共に DFB 62 に送られ、実行ユニットの待ち行列の一つに記憶される (例えば、バックアップチェックポイントは、チェックポイントフィールド 614 に記憶される。図 22 参照)。従って、命令実行が完結すると、シリアル番号及びチェックポイント番号は直ちに、実行ユニット (例えば、FXU、FPU、FXAGU、又は LSU) 内で利用可能な状態となる。

2. 誤予測命令、RED Mode、及び実行トラップ起動によるバックアップ

誤予測命令がウォッチポイント304によって検出されたとき、又はRED Modeへのエントリが行われたときは何時でも、BKUP402によってバックアップサイクルが起動され、実行例外からの回復が行われる。この実行例外からの回復はバックステップだけ、又はバックアップだけ、或いはバックアップと1以上のバックステップとの組合せを必要とする。BKUP402はチェックポイント・ベクトル・ロジックユニットを含み、このユニットは各サイクル毎に、DFB62及びウォッチポイント304から受け取る命令の実行状態に関する情報を評価し、どの命令が誤って予測されたか、又はどの命令が実行例外を起こしたかを決定する。典型的なCPUでは、ロジックユニット404は、DFB62又はウォッチポイント304から受けた各チェックポイントに関して“OR”即ち論理和を取る。最早チェックポイント選択ロジック405は、どのチェックポイントが最早かを決定し、後の障害命令に対応するe-トラップが、既に強制終了させたロジック404によって識別されたチェックポイントへのバックアップを起動しないようにする。BKUP402は、どのチェックポイント番号をバックアップに使うかを決定する際に、誤予測及び実行例外を考慮する。

マシンは、誤予測命令が最早チェックポイントであるときには、誤予測命令チェックポイント間で戻される。RED Mode起動のバックアップに関しては、マシンはコミットしたシリアル番号(CSN)に最も近く、それよりは大きい(Aーリング循環の意味で)チェックポイントへ戻される。また、実行例外に関しては、チェックポイントの選択はより複雑であって、実行例外の性質に依存する。

実行例外(e-トラップ)起動のバックアップの場合、据置及び非据置トラップによって、異なったバックアップチェックポイント

番号の計算が必要になる。通常の非据置トラップに関しては、もしISNがチェックポイント $n+1$ を越えて移動したとき、チェックポイント n の命令 i が障害命令であれば、マシンはISNからチェックポイント $n+1$ 、即ち障害命令の後

のチェックポイントまで戻される。もし、チェックポイント $n+1$ がアクティブでもなく又は有効なチェックポイントでもなければ、チェックポイントを要求する命令は発せられず、又は時間切れチェックポイントも形成されないから、バックアップは行われない。その代わり、障害命令がチェックポイント直後のシリアル番号スロットになれば、ISNから障害命令に後退するバックステップだけを使う回復が適当である。例えば、チェックポイント m がシリアル番号 $SN=10$ を持つように割り当てられ、その時スロット $SN=11$ にある命令が障害命令であれば、チェックポイント $m+1$ へバックアップの代わりに、チェックポイント m へのバックアップが起動される。即ち、マシンは障害命令以前のチェックポイントまで戻される。

実行されたチェックポイント命令が投機的分岐又はその他の制御転送命令に対応している場合には、実行されたチェックポイント命令と次の命令との間に、プログラムカウンタの不連続点が存在することになる。典型的なバックステップ動作は、プログラムカウンタの不連続点を越えて実行されることはなく、単一プログラム命令のシリアル番号差があるだけでも拘わらず使用することは出来ない。この状態では、バックアップ動作によってのみ、プログラムカウンタとマシン状態を適切に回復することができる。

据置トラップは、非据置トラップとは異なるバックアップ・デスティネーション計算を必要とする。例えば、チェックポイント m がシリアル番号 $SN=10$ を持つように割り当てられ、その時 $SN=11$ にある命令が障害命令であって、据置トラップを発生すれば、

ISNはトラッピング命令に先立つ命令よりはむしろトラッピング命令を指定するから、マシンはチェックポイント m の代わりにチェックポイント $m+1$ (もし在れば) まで戻される。もし、その時チェックポイント $m+1$ がなければ、バックアップは起動されない。その代わり、正確な状態は、マシンのバックステップだけを用いて回復される。表10は、チェックポイント番号及び障害命令のシリアル番号に関する幾つかを条件として行った据置及び非据置トラップに対するバックアップ作用の概略を示す表である。

バックトラック305が、マシンバックアップによって影響を受けるCPU51にバックアップが生じていることを知らせる信号と、この影響を受けるCPU51の各要素にバックアップに使用するチェックポイント番号を知らせる4ビットのデータ信号とから成る信号(DO__BACKUP)をアサートしたとき、プロセッサの状態は実行されたチェックポイントのシリアル番号に於いて回復される。実行されたチェックポイントの状態は、マシンに局部的に記憶されるから、CPU51の各々は、DO__BACKUP信号に応答して影響を受けたユニット内の記憶領域から状態を“回復”しなければならない。

表10. チェックポイント番号及び障害命令のシリアル番号を条件として行った据置及び非据置トラップに対するバックアップ作用

トラップは据置か又は非据置か?	トラッピング命令はチェックポイントの直後にSNを持っているか?	チェックポイントn+1は有効か? (チェックポイントnは有効と仮定する)	取られたバックアップアクション
非据置	NO	NO	バックアップなし
非据置	NO	YES	n+1にバックアップ
非据置	YES	NO	nにバックアップ
非据置	YES	YES	nにバックアップ
据置	NO	NO	バックアップなし
据置	NO	YES	n+1にバックアップ
据置	YES	NO	バックアップなし
据置	YES	YES	n+1にバックアップ

実行されたチェックポイントに於ける状態の記憶、保持、回復を受け持つCPU51のユニット、同様にバックアップ動作を順序づけるCPU51のユニット(特にPSU300内のユニット)は、DO__BACKUP信号を受信し、かつそれに応答する。RRF302はDO__BACKSTEP信号を受信するが、DO__BACKUP信号は受信しないことに注意されたい。これは、RRFがCPU51

をバックアップによってではなく、ステップ毎に状態を回復するのに直接必要とされるからである。

更に明確に言えば、図6に於いて、チェックポイント記憶ユニット106は、DO__BAKUP信号によって特定されるチェックポイント番号に対応するエントリ時に、其処に記憶された実行されたチェックポイントのAPC及びNAPCの値を、CHKPT__PC

及びCHKPT__NPC信号の形で出力する。これらの値はAPC及びNAPCのレジスタ112、113に記憶される。同様に、制御レジスタチェックポイント記憶ユニット817は、DO__BAKUP信号によって特定されるチェックポイントに関して実行された制御レジスタの内容を、CHKPT__CNTRL__REG信号の形で出力し、これら内容は図17に図示の制御レジスタファイル800中の対応する制御レジスタNORD/WR更新ロジックによって記憶される。更に、図16に於いて、FXRFRN604、CCRFRN601、及びFPRFRN605各々のチェックポイント記憶ユニット616は、DO__BAKUP信号によって特定されるチェックポイントに関して実行されたチェックポイントのリネームマップを、CHKPT__MAPの形の信号で出力する。この実行されたチェックポイントのリネームマップは、制御ロジック613によってリネームマッピングロジック615に戻して記憶される。更に、フリーリストユニット700は、FXRFRN604、CCRFRN601、及びFPRFRN605各々の物理的レジスタの実行されたチェックポイントのフリーリストを、CHKPT__FREELIST信号の形で出力し、この信号はフリーリストロジック701に記憶される。

典型的なCPU51では、e-トラップがまだ続いているときでも、バックアップは起動される。e-トラップ例外処理は多重サイクルを必要とするから、マシンが同期(ISN=CSN=RRP)され、トラップを取って処理する以前に、一つ以上のe-トラップをDFB62がアサートすることは可能である。典型的なCPU51の一実施例では、スロット0及び1の各々に関するDFB62からのエラー及び状態信号(ERR__STAT)の各々は、1ビットエラー信号、

6ビットのシリアル番号信号 (FX__SN、FP__S

N、LS__SN、及びFXAG__SN)、及び各命令毎に適当なバックアップチェックポイントを計算するのに使用される16ビットの1-ホット・チェックポイント・ベクトル (FXDF__CHKPNT、FPDF__CHKPNT、FXDF__CHKPNT、LSDF__CHKPNT) からなっている。更に、誤予測命令に関する情報信号WP__MISPREDは1-ホットベクトルであって、其処でのビットiのアサートは、i番目のチェックポイントが誤予測命令に対応し、バックアップを必要としたことを示す。チェックポイントベクトルロジックユニット404は、スロット0及び1の各々からの16ビットの1-ホットチェックポイントベクトルの全てについて、累積的に論理和を取る。この動作はマシンがベクトルをトラップテーブルにするまで続けられる。論理和を取ったチェックポイントベクトルは、最早チェックポイントを決める最早チェックポイント選択ロジック405に供給される。たとえ、後の命令に対応して実行例外が生じて、前のバックアップによって既に強制終了したチェックポイントへのバックアップは起動されない。典型的なCPU51では、ウォッチポイント誤予測信号 (WP__MISPRED) は16ビットの1-ホットベクトルで、其処でのビットiのアサートは、i番目のチェックポイントが誤予測命令に対応し、バックアップを必要としたことを示す。

3. 何れかの命令境界へのプロセッサのバックステップ

マシンバックステップはDFB62に於ける障害命令から発生された実行例外 (e-トラップ) の結果としてのみ起動される。マシンバックステップはマシンバックアップとの組合せで起こることが多いが、バックアップを起動しなくても起動される。

バックステップユニット (BKST) 403はバックステップの実施を受け持っている。マシンの“バックステップ”はDFB62

に於ける障害命令から発生するe-トラップの結果として起こるだけである。バックステップが実施されると、2つの事象がマシンに起こる。第1に、未決命令

が強制終了されるため、資源は回復されて利用可能になる。これら資源にはシリアル番号、チェックポイント、ウォッチポイント、及び割り当てられた物理的レジスタ及び論理レジスタが含まれる。第2に、ISNはバックステップ動作中減分するが、CSN及びRRPを下回って減分することはない。障害命令は依然としてそのアクティブビットをA-リングセット中に有していて、それによってCSN及びRRPの送り(advancement)を抑えている。障害命令がエラーを起こさずに首尾良く実行されたときだけ、その命令に対応するA-リングビットがクリアされる。(据置トラップを上手く扱った時に、A-ビット(及びB-ビット)はクリアとなって、CSN(及びNMCSN)の送り及び資源の最終回復を許容することを除く)。

マシンバックステップ動作は、バックステップ始動制御信号(BACKSTEP_LAUNCH)及びバックステップに使用したい命令のバックステップシリアル番号を識別するPLSM307からのバックステップシリアル番号識別信号(BACKSTEP_TO_SN)によってBKST403に於いて開始される。これに応じて、BKST403は、バックトラック305がバックステップモードを開始していることをICRU301及び他のCPU512知らせるバックステップ通知信号(DO_BACKSTEP)を発生することによってバックステップ手続きを順次実施する。BKST403は、バックステップを終えた命令番号のカウントを識別し、PLSM307によるタスクを果たすために起動することが可能な周辺制御装置として見る事ができる信号を発生する。BKST703はPLSMの要求に応じてタスクを開始し、タスクを果たし終え

た時には、タスクの終了をPLSMに指示して、次の要求に待って待機する。

BKST403は2つの条件形式に関するバックステップを扱う。その第1の形式は非据置e-ートラップである。非据置e-ートラップは、据置トラップ信号がPLSM307によってアサートされず(ISA_DEFERRED_TRAP=0)、かつバックステップ開始制御信号が、据置トラップではないe-ートラップを示す信号(BACKSTEP_LAUNCH=1)であるとアサートされている場合に起動される。この場合、バックステップ量はバックステップされるシ

リアル番号 (BACKSTEP__SN) と次ぎに発せされるシリアル番号 (NISN) との差を決めるBKST403で計算される。発せられた命令のシリアル番号 (ISN) ではなく、次ぎに発せられるシリアル番号 (NISN) を使用することによって、マシンは障害命令直前の命令まで戻る。このバックステップ量は、CPU51内の記憶領域に保存され、障害命令の前の命令に到達するまで減分される。

第2のバックステップ条件は据置e-トラップを含んでいる。据置e-トラップは、据置トラップ信号がPLSM307によってアサートされてる信号であって (ISA__DEFERRED__TRAP=1)、かつバックステップ開始制御信号が、据置トラップであるe-トラップを示す信号 (BACKSTEP__LAUNCH=1) であることをアサートしている場合に起動される。この場合、バックステップ量はバックステップされるシリアル番号 (BACKSTEP__SN) と、発せされるシリアル番号 (NISN) との差を決めるBKST403で計算される。此処でマシンは障害命令の前の命令ではなく、障害命令にバックステップされる。計算されたバックステップ量は、通常の非据置バックステップより小さい。

典型的なCPU51では、4つの命令の内の最大のものが、各マシンサイクルに関してバックステップされる。従って、4つまでの命令の多重バックステップはそれぞれ行き先 (バックステップの) 命令に到達する。一般に、シングルサイクルに関してバックステップ命令の数を限定する必要はない。典型的な実施例に於けるもう一步進めた限定は、PCをバックステップ量だけISN (又はNISN) から減じても、適当な状態回復とはならないから、バックステップ方法はプログラムカウンタの不連続の原因となる命令に関するバックステップをサポートするものではない言うことである。

“IDLE” 及び “BACKSTEP” という2つの状態からなる簡単なマシン状態は、バックステップを順次実施するのに使用される。IDLE状態にある間の “BACKSTEP__LAUNCH” アサートは、BACKSTEP状態への遷移となる。また、BACKSTEP状態にある間にDO__BACKSTEP

がアサートされると、2つの終了条件が評価される。これら2つの条件の何れかが、BACKSTEPを終了させ、IDLEに戻す。

バックステップ中に於けるCPU51の状態回復には、マシンレジスタ資源を更新すること、及びプログラムカウンタをバックステップ量だけ減分することだけが含まれる。上記のように、構成上の制御レジスタを修正する命令は、マシンを同期するか、チェックポイントを実行するから、バックステップ中に構成制御レジスタの回復を必要としない。典型的なCPU51では、以下のマシン状態はバックステップ中に更新される。即ち、プログラムカウンタ(PC)、次のPC、RRFに記憶された情報、及びリネームマップに記憶された情報が更新される。トラップPC、トラップRRF、及びBRB59の特権レジスタを含むその他の制御レジスタは、バックステップ動作によって修正も更新もされない。マシンの資源は、レ

ジスタリクレームフォーマットファイル(RRF)302に記憶されたロジカルツウーオールドーフিজカルマッピングに関連するレジスタファイルから回復される。バックステップ通知信号(DO_BACKSTEP)がアサートされると、ICRU301、RRF302、及び他のCPU512は、バックステップモードが起動されていることが通知される。バックステップに於いて、各バックステップサイクル中に回復及び記憶されるRRF302アイテム数は、バックステップカウント信号によって示されるバックステップされる命令の数に等しい。

従って、これらの状態の記憶、保持、及び回復を果たすCPU51は、バックステップ動作を順次果たすCPU51(特にPSU300内の)と共に、BRB59、ISB61、及びDFR62内のユニットを含めて、バックステップ信号を受信し、それに応答する。RRF320は、CPU51をバックステップさせて、バックアップではなくステップ毎にそれを回復することだけに直接関与するから、DO_BACKSTEP信号を受信するがDO_BACKUP信号は受信しない。

レジスタリクレームフォーマットユニット302は、バックステップをアサートした信号(DO_BACKSTEP)に応答し、リクレームされて回復される

RRFアイテムの数は、DO_BACKUP信号内のバックアップステップカウント数に等しい。RRFは、ロジカルツウーオールドフィジカルマップ、即ちソース又は行き先データ値を保持するロジカルツウーオールドフィジカルレジスタの割当を含んでいる。RRF302の命令に割り当てられたレジスタ資源、又はレジスタマップの何れもCSN又はRRPを進めることによってフリーリスト700へは自由に戻れないから、レジスタの実際のデータは妨害されない。RRFは、バックステッ

プした命令をもと実行したきと同じ関係で、フィジカルレジスタ（オールドフィジカルレジスタ）をロジカルレジスタを再度組み合わせる。レジスタマッピングを一つずつ回復することによって、各命令のレジスタ状態が回復される。要するに、レジスタ資源から見れば、マシンはバックステップ動作中後退する。図18はRRFノブロック図である。図14は、ロジカルフィジカルマッピングが、レジスタリネームファイルFXFRN、CCFRN、FCCFRN及びFPRFRNに回復される仕方を示す図である。

RRFの典型的な実施例では、幾つかの情報（LOG_OPHYS_TAGS）を含むロジカルオールドフィジカルレジスタ情報は、データ保存構造又はユニット366の何れかに保存のため、読み／書き制御ユニット365にまとめられている。回復のためRRFをレジスタリネームマップに読み出すとき、RRFデータは回復のためにもとのフォーマットに戻される。本願の教示から当業者は、種々の保存フォーマット及びデータのまとめ／解き方の手順をRRFデータに適用して保存領域を最小化し得ることが分かるだろう。

特に、バックステップが行われているとき、読み／書きロジック365は、保存ユニット366を制御して、DO_BACKST信号によって識別される命令に対応するロジカルオールドフィジカルレジスタ・タグ・マッピングをLOG_OPHYS_TAGS__BKST信号の形で出力する。図8、16、18、及び39に於いて、ロジカルオールドフィジカルレジスタ・タグ・マッピングは、FXRFRN640、FPRFRN606、及びCCRFRN610に復元される。図16に示すように、DO_BACKST信号に応答する制御ロジック6

13は、これらのマッピングをリネームマッピングロジック615に復元する。

更に練られたバックステップ計画は追加情報を維持することによって実施される。しかし、本願発明による方法では、プログラムカウンタの値は基本ブロック内の命令に対応する。即ちバックステップされる命令の間にはプログラムカウンタの中には不連続が存在しないから、方法は簡素化される。バックステップユニット403は単に実行された減入れのシリアル番号からのバックステップ数、又はバックアップなしにバックステップを用いられたときの現在のISNからバックステップ数を信号で合図する。もし、プログラムの不連続があれば、それらプログラムカウンタの減分は、必ずしも適当な回復とはならない。しかし、本願に含まれている教示にから当業者は、より複雑なバックステップ技術を用いることによって、プログラムカウンタの不連続が許容されることが分かるであろう。

2つのバクトラック機構、即ちマシンバックアップ及びマシンバックステップは、投機的命令の続発を取り消す効果的な方法及び構成でプリサイズ・ステート・ユニット(Precise State Unit)を提供する。マシンバックアップは、プリサイズ・ステート・ユニットがCPU51をアクティブチェックポイント境界に対応する状態にすることを可能にする。マシンバックステップは、プリサイズ・ステート・ユニットがCPU51をチェックポイント境界の間の特定の命令境界にバクトラックすることを可能にする。バックアップは、マシン状態回復のための粗いけれども早い方法及び構成を提供し、バックステップはマシン状態回復のための正確だが遅い方法及び構成を提供する。図41は何れの命令境界に於いても、正確な状態を保持し、回復するための本発明による実施例の流れ図を示す。図42はロジカル-フィジカルマッピングを、レジスターリネームファイルに復元する方法を示す図である。

図43は2つのバックステップを伴うマシンバックアップの例を

示す図である。バックステップは一度の幾つかの命令を、単一ステップ又は多重ステップとして実行できる。この例では、CPU51は命令障害が検出された時点で、ポインタISNに対応する命令を既に発している。マシンは障害命令の後

の最も近いチェックポイントにマシンを戻すことによって応答する。このもっと近いチェックポイントとISNとの間の命令は強制終了させられる。4つの命令の量による第1のバックステップは、それらの命令を発している間、4つの命令（第1バックステップ参照）に関わる全ての資源を再生することによって、マシンを障害命令に更に近づける。次いで、3つの命令量による第2（及び最終）バックステップは、それら残りの命令（第2バックステップ領域参照）に関わる全ての資源を再生することによって、障害命令の直前にマシンを持ってくる。

チェックポイントまでの戻りは早い、チェックポイントを回復するのに多くのマシン資源が必要になる。誤予測命令は全て、チェックポイントが作られる要因となるから、状態はワンステップで誤予測命令に戻される。バックステッピングはバックアップより遅いが、本発明のバックステップ手続きはマシン状態を、障害命令前のチェックポイントに於ける状態にではなく、特定命令の実行点に在ったときの状態に戻す。そして、本発明のバックステップ構成及び方法によって、マシンは従来の方法によるように、チェックポイントの前のチェックポイントへ行き、それから障害命令にバックステップするのではなく、障害命令の後のチェックポイントに戻ることが出来る。

D. 例外及び誤予測回復時に於ける優先ロジック及び状態マシン動作

図44はCPU51内に機能的に置かれた典型的な優先ロジック及び状態マシン(PLSM)307を示す。PLMSは幾つかの役

割を持っている。第1に、未処理又は並行例外群の最早の例外を識別する役割を果たす。例外条件には、RED Mode、発行トラップ(i-traps)、実行トラップ(e-traps)、割り込み、及び誤予測が含まれる。第2には、マシンサイクル中に到着する例外及び誤予測を、相互間で、また並行処理される例外及び処理待ちのものとの間の優先順位を決める役割を果たす。第3に、形式の異なる例外及び誤予測間の可能な状態遷移を決定する役割を果たす。第4には、Backtracking (Backupそして/又はBackstep)、REDモード処理を起動する制御信号を発する役割を果たす。

命令の発行又は実行中に起こる種々の形式の例外について、PLSM307に

よる例外及び誤予測の優先順位付け及び扱いがよく分かるように説明する。REDトラップは、種々の条件下で起こり、全てのトラップの中で最も高い優先順位を有する非同期トラップである。RED Modeは、SPARC V-9アーキテクチャマニュアルに説明されている。i-trapsは命令発行時にISU200によって発生される発行トラップである。これらは、何れのレジスタスピル/フィル(spill/fill)トラップ、何れのトラップ命令(TA、Tcc)、命令アクセスエラー、違法命令、特権演算コード等を含んでいる。一般に、発行トラップは、取り扱いに関して低い優先順位にある割り込みに較べれば、高い優先順位にある。信号を送られるその他の全てのトラップ(誤予測、e-トラップ又はredトラップ)は、一時的には発行命令よりは早い命令に向けられる。

e-trapsはデータフォワードバスの一つから信号を送られる実行トラップである。実行トラップは、SN順に他の実行トラップに対して優先順位を付けなければならない。これは、実行トラッ

プは必ずしもチェックポイント境界で発生するとは限らず、また障害命令に達するにはバックステップが必要になる場合があるからである。もし、マシンが障害命令に達するためにバックアップを実施すれば、何れかの新たな例外がより早い命令に向けられる。もし、実行トラップが順次実施されていれば、何れか新たな誤予測がより高位の優先順位となる。これは、誤予測命令は常にそれが利用し得るチェックポイントを有しているからである。もし、PSU300が必ずしも何れかの例外を実施していなければ、実行トラップ及び誤予測は、それぞれ他に対して優先順位が付けられていなければならない。割り込みは、非同期トラップのもう一つの形である。割り込みは他の全ての例外より下位の優先順位にある。

投機的に発した転送制御命令が誤予測であることをウォッチポイントが検出したとき、ウォッチポイントユニット304は誤予測に信号を送る。もし、何れかの非RED例外が順次実行されていれば、誤予測は常にチェックポイント境界上にあつて、一時的に早い命令に対して起こるから、優先順位を付ける必要はない。もし、例外が実施されていなければ、誤予測は割り込み及び発行トラップより高位の優先順位を持つが、実行トラップに対しては優先順位を付けなければなら

ない。

最早シリアル番号選択ロジックユニット (ESNSL) 481は、DFB62の各実行ユニットから、実行完了した命令のシリアル番号及び実行中に例外が起こったかどうかに関する情報、及びもし起こっていたら、その例外の形に関する情報を含む状態情報を受けると共に、ウォッチングユニット304からはそのサイクル中に起きた何れかの誤予測を示す信号 (WP_MISPREDICT) を受け、更に現在のISNを受け、また更にESNSL481からは、最終マシンサイクル (CURR_EARLIEST_SN) から

の新たな例外情報を受け取る前の現マシンサイクルで処理された例外のSNを示す帰還信号を受け取る。各実行ユニットからの信号はERR_STATを含んでいる。例えば、もし64ロケーションA-リング312の命令SN=10及びSN56に対して例外が生じれば、最早例外はISNの値を知ることなしには決めることは出来ない。もし、ISN=40なら、SN=56が最早例外となる。しかし、もしISN=6なら、SN=10が最早例外となる。

ESNSLU481は、最早例外及び誤予測を優先順位及び実行スイッチングロジック (PESL) ユニット482に知らせて、PESLが最早SN基準及び例外の形に基づいて例外及び誤予測処理に優先順位を付けることが出来るようにする。また、PESLはISU200から発行トラップ信号 (ISU_ITRAP) 及び割り込み信号 (ISU_INTERRUPT_PEND) を受信すると共に、Backtrackがチェックポイント情報の比較に基づいて誤予測が実行トラップより早く生じたことを決定する場合には、Backtrackユニット305からMISPREDICTED_EARLIER信号を受信する。また、PESLは状態マシン及び制御ロジック (SMCL) 483から状態信号を受信する。SMCLは状態マシン484及び例外制御信号発生ロジック484を含み、現在処理している例外 (例えば、実行トラップ、発行トラップ、RED等) があればその形を識別する。例外の状態は状態マシン484に記憶される。PESL482はこれらの入力を現在及び未決の例外の優先順位を決めるのに使用し、現在実施中のマシンサイクルのための新たな例外に形を識別する信号 (NEW

__EXCEPTION__TYPE)を発生する。

例外制御信号発生ロジック484は、DPU51内の他のユニットにCPUの状態を知らせると共に、場合によっては、状態に応じ

て他のCPUユニットにアクションを取らせる信号を発生する役割を持っている。更に明確に言えば、ECSIGはTAKE__TRAP信号を発生し、それをトラップが行われる制御部分及びトラップの種類を含むトラップスタックユニットに送る。このトラップスタックユニットはDEFERRED__TRAP__INCを、据置トラップが起きたかどうかを示すICRU301に送り、例外を越えて一つだけCSN及びNMCSNを進ませる。また、ECSIGはBacktrackに対して、どの命令SNにバックステップするか(BACKSTEP__TO__SN)、REDモード例外があるか、そしてREDモード例外の場合にはどの命令にバックアップするか(BACKUP__RED)を告げる幾つの信号をバックトラックユニット305に送る。また、ECSIG484は命令強制終了信号をDFB62(KILL__ALL)及びISU200(ISSUE__KILL)に送り、DFB62に対しては、実行待ち行列中の命令の実行を強制終了するように指示し、ISUに対しては、そのマシンサイクル中に扱われている例外又は誤予測に適当な命令の発行を中止するように指示する。

PLSM307はトラップを探知して5つの異なる形、即ち割り込み、レッドトラップ、発行トラップ、実行トラップ、誤予測に区分する。どの例外が現在順次処理されているかによって、検知される他の例外は高い優先順位を持っているか、そうでない場合もある。もし、新たな例外が高い優先順位を持っていれば、PESL482はSMCL483に対し現在の例外を順次処理することから新たな高い優先順位の例外又は誤予測にスイッチするように指示する。状態マシン484に於けるこれらの状態遷移は、現在の例外の形、即ち新たなエラーがウォッチポイント、ISU、又は6つのデータフォワードバスの何れから報告されたか、及び新たな例外が現在の

ものより一時的に早いかどうかによる。

可能な状態遷移は以下のようにして起こる。RED-Notラップ形は状態REDからの遷移を強制する。ITRAP-3ラップ形（誤予測、実行トラップ、REDOアラート）は他の一つの状態への遷移を強制できるが、割り込みは遷移を強制しない。ETRAP-Redアラート及び誤予測は遷移を強制し、常に高い優先順位を有し、他の実行トラップは優先順位化を必要とし、割り込みは低い優先順位を有し、マシンはマシン同期のためバックアップ又は待機状態の何れかにあり、発行トラップを発する命令は発せられないから、発行トラップは不可能である。INTERRUPT-全トラップ形は高い優先順位を有し、遷移を強制することが出来る。MISPREDICT-3ラップ形（誤予測、事項とラップ、レッドアラート）は高い優先順位を有し、優先順位付けを必要とせず、張り込みは低い優先順位を有し、命令発行は誤予測処理ちゅう強制終了させられているので、発行トラップは起こらない。IDLE-Redトラップは最高優先順位を有し、同時誤予測及び実行トラップが、チェックポイントの比較に基づいて誤予測が早く起きたか、実行トラップが早かったかの決定が行われるBACKUPへの遷移に帰着する4つの形の一つを伴う。Backtrackは優先順位より早く決定が出来るから、優先順位化に基づいたチェックポイントに対する据置は好ましい。しかし、優先順位化はPLSMに於いて実行が可能である。最後に、発行トラップ、次いで割り込みが順次処理される。

E. トラップ・スタックによるトラップの処理

先に示した如く、トラップは図5の命令パイプライン内で幾度も生じ、且つ、トラップ処理ルーチンにより処理されねばならない（即ち、受入れられねばならない）。図17を参照すると、トラップ

を受入れるべき例外をPSU 300が既述の手法で検知すると、PLSM 307はTAKE_TRAP信号を生成し、これにより、制御レジスタファイル800にトラップを受入れさせると共に、受入れられるべきトラップのタイプを識別する。これに応じ、リード/ライト/更新ロジック816はTBAレジスタ812から対応トラップベクトル(TRAP_VEC)信号を読み出してこれをBRB 59に送る。図6を参照すると、PCロジック106はTRAP_VEC信号を使用して新たなFPC、APC およびNAPCを算出し、これによ

り、対応トラップ処理ルーチンの命令を取り出して出力する。

トラップ処理ルーチン内で生じたトラップがSPARC-V9アーキテクチャにより特定された如きネスト手法（即ち、受入れられたトラップの内部でトラップが受入れられる手法）で処理される様に、制御レジスタファイル800 は図17に示されたトラップスタックユニット815 を含んでいる。このトラップスタックユニット815 は図45に更に詳細に示されているが、該ユニットは、記憶エレメントもしくは記憶エントリを備えたデータ記憶構造であるトラップスタック820 を含んでいる。

記憶エントリの各々は4個のフィールドを有しており、これらのフィールドはSPARC-V9アーキテクチャマニュアルによれば、トラップ・プログラム・カウンタ(TPC)、トラップ・ネクスト・プログラム・カウンタ(TNPC)、トラップ・ステート(TSTATE)、トラップ・タイプ(TT)の各フィールドである。TPC およびTNPCフィールドは、トラップが受入れられる時点の、BRB 59のAPC レジスタ113 およびNAPCレジスタ114 内の夫々のAPC 値およびNAPC値を含んでいる。SPARC-V9アーキテクチャマニュアルに記載された様に、TSTATEフィールドは、トラップが受入れられた時点で図39に示されたCCRFRN 610内のXICCレジスタの内容と、図17に示されたASI レジスタ808、

CWP レジスタ809 およびPSTATEレジスタ805 の内容とを含んでいる。TTフィールドは、受入れられたトラップのタイプを識別する。

図45に再度戻ると、トラップが受入れられたとき、現在のPC、NPC、TSTATE およびTTの各値は、トラップスタック820 の記憶エントリの内のひとつの記憶エントリの対応フィールド内に記憶されている。受入れられたトラップのネスト数を表示するトラップレベル(TL)は次にインクリメントされると共に、トラップ処理ルーチンは受入れられたトラップの処理を開始する。しかし乍ら、もしトラップ処理ルーチンの間にトラップが生じたときには、この第2のトラップの時点におけるPC、NPC、TSTATEおよびTTの各値は、トラップスタック820 の記憶エントリの内のもうひとつの記憶エントリの対応フィールド内に記憶される。次に、第2トラップを処理する為に、第2のトラップ処理ルーチンが呼び出される。第2

トラップ処理ルーチンの終わりにてはDONEもしくはRETRY 命令が実行されて、第2トラップに対する記憶エントリ内に記憶されたPC、NPC、TSTATEおよびTTの各値が対応レジスタ内に書込まれると共にトラップレベルがデクリメントされる。この結果、CPU 51は、第2のトラップを受入れた時点の状態に戻ると共に、第1トラップ処理ルーチンが第1トラップの処理を再開する。その後、第1トラップ処理ルーチンのDONEもしくはRETRY 命令が実行されたとき、CPU 51は第1トラップを受入れた最初の状態に戻る。

以上から、SPARC-V9アーキテクチャにより必要とされる如く、トラップスタックユニット815 は多数のトラップレベルを含むことにより多段階にネストされたトラップの処理をサポートすることは明らかである。但し、SPARC-V9アーキテクチャが4段階のトラップレベルをサポートすべく4個の記憶エントリを備えたトラップスタックを指定してはいるが、上記トラップスタック820 は4段の所要の

トラップレベルをサポートすべく8個の記憶エントリ0～7を含んでいる。付加的な4個の記憶エントリは、レジスタまたは記憶エレメントの名称変更の概念が以下に記述する如くトラップスタックユニット815 まで拡張され、トラップが投機的に受取られ且つそこから戻れる様になる。更に、前述の如く、記憶エントリ数がトラップレベル数より大きい限り、本発明は任意の個数のトラップレベルにて実施され得る。

トラップスタックユニット815 はフリーリストロジック821 を含み、該ロジックは、各トラップと連携されたTPC、TNPC、TSTATEおよびTTの値を記憶すべく現在使用可能なトラップスタック820 の記憶エントリの全てのリストを記憶している。図46を参照するに、フリーリストレジスタ822 は、各マシンサイクルの間に、8ビットのFREELISTベクトル信号形態とされた現在使用可能な記憶エントリのリストを記憶する。このFREELIST信号の各ビットは、トラップスタック820 の記憶エントリのひとつに対応すると共に、その記憶エントリが現在使用可能であるか否かを識別するものである。FREELISTベクトルにより使用可能であると識別された最初の記憶エントリに対し、ファーストワンエンコーダ823 は次にそのベクトルをWR_ENTRY 信号にエンコードするが、該信号は、この記憶エントリを識

別すると共に、受入れるべき次のトラップに対するTPC、TNPC、TSTATEおよびTTの各値を書込むためのポインタの役割を果たすものである。

FREELISTロジック821 はまた、オールゼロ検出器836 も含んでいる。それは、FREELIST信号がオールゼロビットを含むことによりトラップスタック820 内の記憶エントリはいずれも利用できないことを示す時点を決断する。もしこの場合が生ずれば、それはNO_ENTRY_AVAIL信号を提示する。

図45を参照すると、PSU 300 によりトラップを受入れるべきことが決定され乍らもNO_ENTRY_AVAIL信号は利用できる記憶エレメントが存在しないことを示しているとき、それが出力したTAKE_TRAP 信号は、NO_ENTRY_AVAIL信号により記憶エレメントが利用可能であることが示されるまで、トラップを受入れるべきことを示さない。更に、PSU 300 はISSUE_KILL信号およびFETCH_SHUTDOWN信号を発し、ISU 200 による命令の出力とBRB 59による命令のフェッチとを停止するが、これは、NO_ENTRY_AVAIL信号により記憶エントリが使用可能であることが示されるまで継続される。これによれば、直ちに明らかとなる如く、CPU 51を同期することによりチェックポイントが解除され且つトラップスタック820 内の記憶エントリが回復されて再利用出来るようになる、という効果が得られる。

しかし乍ら、NO_ENTRY_AVAIL信号が発せられないときにPSU 300 は制御レジスタファイル800 に対してTAKE_TRAP 信号を出力するが、該TAKE_TRAP 信号は前述の如く、トラップを受入れるべきことを示す信号とトラップのタイプを識別する信号とを含むものである。これに応じ、制御レジスタ・リード/ライト/更新ロジック816 は、ASI レジスタ808、CWP レジスタ809 およびPSTATEレジスタ805 の内容を読み出してトラップスタックユニット815 のトラップスタック820 に供与する。

これと同時に、トラップスタック820 は、該トラップスタック820 のTPC およびTNPCフィールドに書込まれるべき現在のAPC およびNAPC値を含むWR_TPC_TNPC 信号を受信する。また、後述する様に、トラップスタック820 は、利用できる様になったときに図8のFSR/CCRFRN 610内のXICCレジスタのXICCデータを受信する。

TAKE_TRAP 信号に応じ、トラップスタックRD/WR 制御ロジック822 はTAKE_TRAP 信号により含まれたトラップタイプ (TT) フィールド

ドを抽出すると共に、それをトラップスタック820 に供与する。次に、RD/WR ロジック822 は、受信したAPC、NAPC、ASI、CWP、PSTATEおよびTTの各値を、WR_ENTRY信号により指示された記憶エレメントの適切なフィールド内に直ちに書込む。更に、後述の如く、XICCデータは同一の記憶エレメントのTSTATEフィールド内に書込まれるが、これは、論理的なXICCレジスタの内容が利用できるようになったときのWR_ENTRY_XICC 信号に応じて行われる。

図17を参照すると、制御レジスタRD/WR/UPDATEロジック816 は次にTLレジスタ811 からの旧TL値を受信してそれをインクリメントする。それは次に新たなTL値をTLレジスタ811に書き戻す。

図45に示された如く、トラップ・スタック・リネーム・マッピング(RENAME MAP)・ロジック824 は、新たなTL値、および、TAKE_TRAP およびWR_ENTRY信号を受信する。図47を参照すると、RENAME MAPロジック824 は、夫々が4個のトラップレベルのひとつに対応するTL1-4(トラップレベル1-4)書込マルチプレクサ826~829 およびTL1-4 レジスタ830~833 を含んでいる。新たなTL値およびTL_TRAP 信号に応じ、RENAME MAPロジック824 の制御回路825 はWR_MUX_CNTRL信号によりTL1-4 書込マルチプレクサ826~829 を制御し、これにより、WR_ENTRY信号は新たなTL値により識別されるトラップレベルに対応するレジスタ内に記憶され、且つ、その他のレジスタは以前のマシンサイクルにおいて記憶したのと同じ値を再記憶する。

従って、レジスタのひとつに記憶されたWR_ENTRY信号により識別される記憶エントリは、RENAME MAPロジック824 内の新たなTL値により識別される現在のトラップレベルにマッピングされる。更に、他のレジスタ内に記憶された信号により識別される記憶エントリは、以前のマシンサイクルにおけるのと同様にしてマッピングが維持

される。

図46を参照すると、FREELIST 821のファーストワンエンコーダ823はTAKE_TRAP信号も受信する。TAKE_TRAP信号がトラップを受入れるべきことを示すとき、ファーストワンエンコーダ823はOR回路835に対し、WR_ENTRY信号に対応するビットが“0”とされた8ビット信号を送り返す。OR回路835はまた、(図45に示された)RRFロジック837からの8ビットFREE_ENTRY信号を受信するが、そのビットは、トラップスタック820の記憶エントリの中で前回マシンサイクルで回復されて現在使用可能となった記憶エントリを識別するものである。バックアップの間、OR回路835はAND回路836から8ビットのBACKUP_FREELIST信号を受信するが、該信号のビットは、バックアップされたチェックポイントが作られた時点において使用可能であった記憶エントリを識別するものである。ファーストワンエンコーダ823から受信した信号のビットは、FREE_ENTRYおよびBACKUP_FREELIST信号と論理和が取られており、且つ、最終的なFREE_LIST信号はレジスタ822に供与される。従って、バックアップを除けば、次のマシンサイクルにおけるFREELIST信号は、WR_ENTRY信号内で“0”に設定されたビットに対応する記憶エレメントはもう使用出来ないことを示している。

図7を再度参照すると、PSU 300はTAKE_TRAP信号を生成すると同時に、CHKPT_T_REQ信号により、ISU 200に対し、トラップ処理ルーチンから最初に発せられた命令に対してチェックポイントを割当ててることを要求する。これに応じ、ISU 20は、制御レジスタファイル800を含め、CPU 51内の種々のブロックおよびユニットにDO_CHKPT信号を出力する。

図47に戻り、RENAME_MAP_1-4信号はTL1-4レジスタ830～833により出力されることから、夫々、トラップレベル1～4に対応し

ている。更に、これらの信号はTL1-4レジスタ830～833により出力されることから、それらはトラップレベル1～4に対する記憶エントリの現在のマッピングを与えるものである。

しかし乍ら、前述の如く、トラップが受入れられたとき、RENAME_MAPロジック824は新たに利用し得る記憶エントリを新たなトラップレベル(TL)にマッピングする。但し、このトラップレベルに対する別の記憶エントリの旧マッピングは、

前述の如くバックアップが生ずるときには記録しておかねばならない。従って、図48を参照すると、PRF ロジック837 は、自身がRENAME MAPロジック824 から受信したRENAME_MAP_1-4信号からの旧マッピングを、それらが不要となるまで保存しておく。

トラップが受入れられたとき、マルチプレクサ843 は、新たなトラップに対するTL（即ち、旧TLは1だけインクリメントされる）に対応するRENAME_MAP信号を出力する。換言すると、このマルチプレクサは、次のマシンサイクルにおいて新たなマッピングを反映するRENAME_MAP信号のみを選択する。この信号は次に、デコーダ842 により、記憶エントリの内のいずれが新たなマッピングにより新たなトラップレベルに対して置換えられているのかを示す8ビット信号にデコードされる。ここで、TAKE_TRAP 信号はトラップが受入れられつつあることを示すことから、デコードされた信号は、RRF ロジック837 のRRF 記憶ユニット839 に対してAND 回路841 を介して供与される。しかし乍ら、何らのトラップも受入れられていない場合、AND 回路841 はRRF 記憶ユニット839 に対して全てのビットが“0”に設定された8ビット信号を供与する。

RRF 記憶ユニット839 は、16個の記憶エレメントすなわちエントリを含んでいる。各記憶エントリは、前述の16個のチェックポイント番号のひとつに対応している。従って、DO_CHKPT信号により

チェックポイントが形成されるべきことが示されたとき、RRF RD/WR 制御ロジック840 はAND 回路841 から受信したデータを、DO_CHKPT信号により識別されたチェックポイント番号に対応する記憶エントリ内に書込んでいる。

従って、RRF 837 は、トラップスタックユニット820 の各記憶エントリの不稼働リストを保持するが、該記憶エントリは、RENAME MAPロジック824 の現在のマッピングによりトラップレベルの内のひとつに現在マッピングされてはいないがトラップレベルのひとつにマッピングする為には依然として使用され得ないものである、と言うのも、トラップレベルに対するその旧マッピングはチェックポイントをバックアップする場合には再記録する必要があるからである。この場合、記憶エントリは不稼働リスト内に保持されるが、これは、形成されるべき

チェックポイントを引起こしたトラップのトラップレベルに対して記憶エントリがマッピングされた時点において形成されたチェックポイントが解除されるまで続くことになる。

RRF RD/WR 制御ロジック840 はまた、PSU 300 から16ビットのCHKPT_CLR 信号も受信する。各ビットは、チェックポイント番号のひとつに対応すると共に、対応するチェックポイントが前述の手法で解除（即ちクリア）されたか否かを示している。この信号に応じ、PRF RD/WR ロジックは、RRF 記憶ユニット839 の対応記憶エントリからデータを読み出す。CHKPT_CLR 信号はクリアされた多数のチェックポイントを識別し得ることから、対応する記憶エントリから読み出されたデータの各ビットは論理和が取られると共に結合されてFREE_ENTRY信号を形成し、それは8ビットのFREE_ENTRY信号として図46のFREELISTロジック821へ送られる。

トラップが受入れられたときにCHKPT_CLR 信号により識別された

チェックポイントが形成された場合、FREE_ENTRY信号により識別された記憶エントリは回復されて現在は再び使用可能となっている、と言うのも、対応するチェックポイントは解除されているからである。その結果、このトラップはもはや行い得ないものではない。従って、バックアップが生じなかったと仮定すれば、FREE_ENTRY信号の各ビットは次に、ファーストワンエンコーダ823により供与された信号の対応ビットと論理和が取られ、次のマシンサイクルに対するFREELIST信号が提供される。

また、トラップが受入れられたときにCHKPT_CLR 信号により識別されたチェックポイントが形成されなかった場合、対応する記憶エレメントから読み出されたデータのビットは全て“0”であり、従って、FREELIST信号の生成に関して何らの影響を有さない。

図45に示された様に、トラップスタックユニット815 はまた、トラップ・スタック・チェックポイント記憶ユニット845 も含んでいる。この記憶ユニット845 は図48に更に詳細に示す様にデータ記憶構造846 を含むが、該構造は、夫々が5個のフィールドを有する16個のアドレッシング可能な記憶エレメントすなわ

ちエントリを有している。

この記憶ユニットは、FREELISTロジック821からのFREELIST信号およびRENAME_MAPロジック824からのRENAME_MAP信号を受信する。而して、チェックポイントを形成すべきことがDO_CHKPT信号により示された場合、トラップスタック記憶RD/WR制御ロジック847は、FREELIST信号およびRENAME_MAP信号を、DO_CHKPT信号により識別されたチェックポイント番号に対応する記憶エントリの適切なフィールド内に書込む。現在のFREELIST信号およびRENAME_MAP信号はチェックポイントが形成されたときに記憶されていることから、トラップレベルに対するトラップ記憶エントリの現在のマッピング、およ

び、利用し得るトラップスタック記憶エントリの現在のリストには、チェックポイントが割当てられている。

チェックポイントに対するバックアップが為されつつあることがDO_BACKUP信号により表されるとき、RD/WR制御ロジック847は、DO_BACKUP信号により識別されたチェックポイント番号に対応する記憶エレメントから、チェックポイントが割当てられたFREELIST信号およびRENAME_MAP信号を読み出す。この点、FREELIST信号はBACKUP_FREELIST信号として読み出される一方、RENAME_MAP信号はBACKUP_MAP信号として読み出される。

図46に示された如く、フリーリストロジック821はBACKUP_FREELIST信号およびDO_BACKUP信号を受信する。DO_BACKUP信号はバックアップが生じていることを表すことから、AND回路838はBACKUP_FREELISTをOR回路835に供与する。前述の如く、ファーストワンエンコーダ823から受信された信号の各ビットは、FREE_ENTRY信号およびBACKUP_FREELIST信号の対応ビットと論理和が取られ、FREELIST信号を生成している。

図47を参照するに、RENAME_MAPロジック824はBACKUP_MAP信号およびDO_BACKUP信号を受信する。DO_BACKUP信号はバックアップが生じていることを表すことから、制御回路825が生成するWR_MUX_CNTRL信号は、TL1-4レジスタ830～833の夫々に対するBACKUP_MAP_1-4信号を供与する。その結果、チェックポイントが割当てられたときに存在したトラップレベルに対するトラップスタック記憶エ

レメントのマッピングは、元通りの状態にされている。

前述の如く、CPU 51は投機的に (speculatively) 命令を発し且つ実行する。トラップスタックユニット815 は、バックアップの場合に以前のトラップレベルを回復する機構を含んでいることから、CPU 51はトラップを投機的に受取るとともにトラップから投機的に戻

ることができる。

トラップを投機的に受入れると共にトラップから投機的に戻るというCPU の能力は、図51に示されている。図示された如く、最初に、実行手順はトラップレベル0に在り、且つ、トラップレベル1～4はトラップスタックエントリ1～4へ夫々マッピングされている。最初のトラップが受入れられるとき、トラップレベルはトラップレベル1にインクリメントされると共に、チェックポイント1が形成され、チェックポイント1における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられ (checkpointed)、且つ、トラップレベル1の新たなマッピングが記憶エントリ5に対して作られる。新たなチェックポイントはチェックポイント2で作られるが、これは予期されたプログラム制御命令などの命令に対するものであり、且つ、チェックポイント2における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられる。第2のトラップが受入れられると共にトラップレベルはトラップレベル2にインクリメントされ、チェックポイント3が形成され、チェックポイント3における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられ、且つ、トラップレベル2の新たなマッピングが記憶エントリ6に対して作られる。トラップレベル2におけるトラップに対するトラップ処理においてdoneもしくはretry 命令が実行されたとき、トラップレベルはトラップレベル1にデクリメントされると共にチェックポイント4が形成され、かつ、チェックポイント4における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられる。トラップレベル1におけるトラップに対するトラップ処理に戻った後、このトラップハンドラの命令に対してチェックポイント5においてチェックポイントが作られ、且つ、チェックポイント5における記憶

エントリマッピングへのトラップレベルにはチェックポイントが割当てられる。更なるトラップが受入れられたとき、トラップレベルはトラップレベル2にインクリメントされ、チェックポイント6が形成され、チェックポイント6における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられ、且つ、記憶エントリ7に対するトラップレベル2の新たなマッピングが作られる。そして更なるトラップが受入れられたとき、トラップレベルはトラップレベル3にインクリメントされると共にチェックポイント7が形成され、チェックポイント7における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられ、且つ、記憶エントリ0へのトラップレベル3の新たなマッピングが作られる。トラップレベル3におけるトラップに対するトラップハンドラ内でdoneまたはretry 命令が実行されたとき、トラップレベルはトラップレベル2にデクリメントされると共にチェックポイント8が形成され、且つ、チェックポイント8における記憶エントリマッピングへのトラップレベルにはチェックポイントが割当てられる。その後、チェックポイント2が形成されたプログラム制御命令が誤って予測されたと判断されたのであれば、チェックポイント2へのバックアップが生じ、チェックポイント2における記憶エントリマッピングへのトラップレベルが回復される。従って、チェックポイント3および7が形成されたトラップは投機的に受入れ/戻りが行われ、一方、チェックポイント6が形成されたトラップは投機的に受入れられている。

前述の如く、トラップスタックのTSTATEフィールドはXICCフィールドを含み、該XICCフィールドは、トラップが受入れられたとき、図8のFSR/CCRFRN 606内の論理的XICCレジスタとしてマッピングされた物理レジスタの内容（すなわちXICCデータ）を保持している。

固定小数点命令はPC命令を受けずにFXU 601 およびFXAGU 602 により実行され得ることから、これらの内容はトラップが生じた時点で利用され得ない。従って、後の時点にて正しいXICCデータを得る機構が無ければ、PSU 300 はXICCデータがトラップを受入れるべく利用出来るようになるまでまたねばならない。

しかし乍ら、図45を参照すると、トラップスタックユニット815 は斯かる機

構、即ち、XICC捕捉ロジック823 を含んでいる。このXICC捕捉ロジック823 は、FSR/CCRFRN 606から、CC_TAGS_C、CC_DV_C、CC_TAGS_F およびCC_DV_F 信号を受信し、一方、トラップスタック820 はCC_DATA_C およびCC_DATA_F 信号を受信する。

図5-2を参照すると、CC_DATA_C 信号はXICC_DATA_C 信号を含むと共に、CC_DV_C 信号はXICC_DV_C 信号を含んでいる。可能であれば、XICC_DATA_C 信号は、トラップが受入れられたときのマシンサイクルにおける論理XICCレジスタ（即ち、図3-9に示されたCCRFRN 610による論理XICCレジスタとしてマッピングされた物理レジスタ）の現在の内容を含んでいる。XICC_DV_C 信号は、XICC_DATA_C 信号の内容が有効か否か（即ち、論理XICCレジスタとして現在マッピングされている物理レジスタの内容が依然として利用し得るか否か）を表している。

トラップが受入れられた時点において論理XICCレジスタとしてマッピングされた物理レジスタの内容は、その時点で既に利用し得る可能性もある。これが生じた場合、XICC_DV_C 信号は、XICC_DATA_C 信号の内容が有効であることを示し、且つ、PSU 300 から受入れたTAKE_TRAP 信号は、トラップが受入れられつつあることを示す。これに応じ、XICC書込ロジック825は、WR_ENTRY信号により識別されたエントリに対応するWRI_ENTRY_XICC信号を提供する。これが生じた場合、トラップスタックユニット815 のRD/WR 制御ロジック82

2 は、XICC_DATA_C 信号の内容を、WRI_ENTRY_XICC信号により識別されたトラップスタック820 のエントリのTSTATEフィールド内に書込む。

しかし乍ら、XICC_DV_C 信号が、トラップが受入れられた時点のXICC_DATA_C 信号の内容が有効でないことを示すときは、XICC_DATA_C 信号の内容はトラップスタック820 に書込まれない。この場合、トラップスタックユニット815 は、トラップが受入れられた時点で論理XICCレジスタとしてマッピングされた物理レジスタの内容が利用できるようになるまで待たねばならない。図5-3を参照すると、これを行うために、XICC捕捉ロジック823 は、現在整合ロジック826 および後期整合配列ロジック827 を含んでいる。

依然として図5-3を参照するに、CC_TAG_C信号はXICC_TAG_C信号を含むが、該

XICC_TAG_C信号は、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの物理レジスタ・タグを含んでいる。更に、CC_TAG_F信号は、FXU 601 およびFXAGU 602 により夫々発信されたFXU_XICC_TAG_F信号およびFXAGU_XICC_TAG_F信号を含むが、これは、FXU 601 およびFXAGU 602 が、論理XICCレジスタを変更する固定小数点命令を実行するときに行われる。これらの信号は、実行されたこれらの固定小数点命令に対するものであると共にCC_DATA_F 信号のFXU_XICC_DATA_F 信号およびFXAGU_XICC_DATA_F 信号が書込まれるべき論理XICCレジスタとしてマッピングされた物理レジスタの物理レジスタ・タグを含むものである。FXU_XICC_DATA_F 信号およびFXAGU_XICC_DATA_F 信号は夫々、FXU 601 およびFXAGU 602 により発信されたものである。

CC_DV_F 信号は、論理XICCレジスタを変更する固定小数点命令をFXU 601 およびFXAGU 602 が実行するとき該FXU 601 およびFXAGU 602 により夫々発信されたFXU_XICC_TAG_F信号およびFXAGU_XICC_TAG_F信号を含んでいる。これらの信号は、実行された固定小数点命令に対する論理XICCレジスタとしてマッピングされた物理レジスタに対してFXU_XICC_DATA_F 信号およびFXAGU_XICC_DATA_F 信号の内容が有効か否か（即ち、利用できるか否か）を表している。

既述の如く、実際のおよび／または予期されたプログラム命令に基づいて命令は実行され得ることから、発信されたFXU_XICC_DATA_F 信号およびFXAGU_XICC_DATA_F 信号は最終的に、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタに書込まれるべき内容を含むことになる。これは、トラップが受入れられるのと同じのマシンサイクル（即ち、現在のマシンサイクル）もしくは後期のマシンサイクルにて生ずる。

これが同一のマシンサイクルで生じる場合、XICC_TAG_C信号は、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタのタグを現在の識別する。XICC_TAG_C信号により識別されたタグは、現在整合ロジック826 の比較ロジック850 および比較ロジック851 により、FXU_XICC_TAG_F信号およびFXAGU_XICC_TAG_F信号と比較される。もし、整合が生じ、且つ、対応する

FXU_XICC_DV_F 信号またはFXAGU_XICC_DV_F 信号が、対応するFXU_XICC_DATA_F 信号またはFXAGU_XICC_DATA_F 信号の内容が有効であることを示すのであれば、現在整合ロジック826 は、対応するFXU_XICC_CURR_MATCH 信号またはFXAGU_XICC_CURR_MATCH 信号を出力する。この信号は、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの内容が、トラップが受入れられたときと同一のマシンサイクル内で利用し得る様になったことを表す。

図52に戻り、このマシンサイクルの間、TAKE_TRAP 信号は依然としてトラップが受入れられつつあることを表し、且つ、WR_ENTRY 信号は依然としてトラップのトラップレベルにマッピングされたエントリを識別している。従って、FXU_XICC_CURR_MATCH 信号またはFXAGU_XICC_CURR_MATCH 信号に応じ、XICC書込ロジック825 は対応するWR2_ENTRY_XICC信号またはWR3_ENTRY_XICC信号を生成する。この信号は、WR_ENTRY信号により識別されたエントリに対応する。次に、トラップスタックユニット815 のRD/WR 制御ロジック822 は、FXU_XICC_DATA_F 信号またはFXAGU_XICC_DATA_F 信号の内容を、対応するWR2_ENTRY_XICC信号またはWR3_ENTRY_XICC信号により識別されるトラップスタック820 のエントリのTSTATEフィールドのXICCフィールド内に、書込む。

図53に戻ると、既述の如く、トラップが受入れられたときのマシンサイクルの後のマシンサイクルにおいて、発信されたFXU_XICC_DATA_F 信号およびFXAGU_XICC_DATA_F 信号は、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタに書込まれるべき内容を含んでいる可能性がある。この状況に備えるべく、XICC捕捉ロジック823 は、タグ配列記憶ユニット854 を含むタグ配列記憶ユニット827 を含んでいる。

記憶ユニット854 は、8個の記憶エントリを含んでおり、各エントリは、トラップスタック820 の各エントリのひとつに対応している。従って、トラップが受入れられつつあることがTAKE_TRAP 信号により示される都度、RD/WR ロジック855 は、そのときのXICC_TAG_C信号により識別されるタグを、WR_ENTRY信号により識別されるトラップスタック820 のエントリに対応する記憶ユニット854 のエントリに、書込む。

比較配列ロジック852 および853 は、夫々、FXU_XICC_TAG_F信号およびFXAGU_XICC_TAG_F信号により識別されるタグを、記憶ユニット854 内に記憶されたタグの各々と、比較する。その結果、もし、

整合が生じ、且つ、対応するFXU_XICC_DATA_F 信号またはFXAGU_XICC_DATA_F 信号の内容が有効であることが対応するFXU_XICC_DV_F 信号またはFXAGU_XICC_DV_F 信号により表されたのであれば、対応する比較配列ロジック852 または853 は、対応するFXU_XICC_DATA_F 信号またはFXAGU_XICC_DATA_F 信号の内容が書込まれるべきトラップスタック820 のエントリを識別する対応FXU_XICC_LATE_MATCH 信号または対応FXAGU_XICC_LATE_MATCH 信号を、出力する。

図54に戻ると、XICC捕捉ロジックは、XICC用待機ロジック828 を含んでいる。又、レジスタ856 はWAIT_FOR_XICC_VEC 信号を記憶する。このWAIT_FOR_XICC_VEC 信号の各ビットは、トラップスタック820 の各エントリのひとつに対応すると共に、対応するトラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの内容が利用可能になるのを対応エントリが待機しているときに提示される。従って、WAIT_FOR_XICC_VEC ベクトル信号は、対応するトラップが受入れられたときに論理XICCレジスタとしてマッピングされた種々の物理レジスタの内容が利用可能になるのを現在待っているトラップスタック820 の各エントリの全てのリストを提供するものである。

TAKE_TRAP 信号により表されるトラップが受入れられると共に、XICC_DV_C 信号が“論理XICCレジスタとしてマッピングされた物理レジスタの内容を利用することができない”ことを示し、且つ、FXU_XICC_CURR_MATCH 信号およびFXAGU_XICC_CURR_MATCH 信号のいずれもまたこれを示さない、という状態が生じる都度、WR_ENTRY信号は、後期のマシンサイクルにおいてこれらの内容が利用し得る様になるのを待っているエントリを、識別する。これに応じ、このエントリは、XICC用待機ロジック828 により、対応トラップが受入れられたときに論理XICCレジスタとしてマッピングされた種々の物理レ

ジスタの内容が利用し得る様になるのを現在待っているトラップスタック820 の

エントリのリストであってWAIT_FOR_XICC_VEC ベクトル信号により与えられるリスト、に対して加えられる。

しかし乍ら、前述の如く、TAKE_TRAP 信号により示されるごとくトラップが受入れられたとき、論理XICCレジスタとしてマッピングされた物理レジスタがこの時点で利用し得ることがXICC_DV_C 信号により示されるか、あるいは、FXU_XICC_CURR_MATCH 信号またはFXAGU_XICC_CURR_MATCH 信号がこのことを示す可能性もある。これが生じる都度、WR_ENTRY信号により識別されるエントリはXICCロジック828 により、WAIT_FOR_XICC_VEC ベクトル信号により与えられたリストに加えられない。

更に、前述の如く、対応するトラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの内容が後期のマシンサイクルにおいて利用し得る様になったときは常に、FXU_XICC_LATE_MATCH 信号またはFXAGU_XICC_LATE_MATCH 信号は、これらの内容が書込まれるべきトラップスタック820 のエントリを識別する。図5 2に戻ると、この時点でWAIT_FOR_XICC_VEC ベクトル信号は依然としてこのエントリをリストすることから、XICC書込ロジック825 は、このエントリを識別する対応WR2_ENTRY_XICC信号または対応WR3_ENTRY_XICC信号を生成する。その後、トラップスタックユニット815 のRD/WR 制御ロジック822 は、FXU_XICC_DATA_F 信号またはFXAGU_XICC_DATA_F 信号の内容（即ち、対応トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの内容）を、対応するWR2_ENTRY_XICC信号またはWR3_ENTRY_XICC信号により識別されたトラップスタック820 のエントリのTSTATEフィールドのXICCフィールド内に、書込む。次に、図5 4に戻ると、このエントリは、WAIT_FOR_XICC_VEC ベクトル信号により与えられ

たりリストから、XICC待機用ロジック828 により除去される。

従って、受入れられ乍らも未だ戻っていないトラップの各々に対し、XICC捕捉ロジックは、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの内容が利用し得る様になる時点、および、トラップスタック820 内のどのエントリ内にこれらの内容が書込まれるべきなのかを、決定する

ことが出来る。その結果、これらの内容はいずれかのトラップが受入れられる時点では未だ利用できない可能性があるとしても、ネストしたトラップを受入れることが可能となる。換言すると、XICC捕捉ロジック823は、対応トラップステート(TSTATE)データの全てが利用可能になる以前にトラップが受入れられることを許容する機構である。

しかし乍ら、トラップ受取から戻ると同時に、トラップが受入れられたときに論理XICCレジスタとしてマッピングされた物理レジスタの内容は、トラップスタック820の対応エントリ内に既に書込まれていなければならない。これは、RETRYもしくはDONE命令を行うためには、これらの内容が、現在において論理XICCレジスタとしてマッピングされた物理レジスタ内に書き戻されるべきことが必要である、という事実に拠るものである。

図54に示される如く、これが生ずるのを確実にすべく、XICC用待機ロジック828はマルチプレクサ857を含んでいる。現在受入れられるつつあるトラップのトラップレベルに現在マッピングされているトラップスタック820のエントリを識別するRD_ENTRY信号に応じ、上記マルチプレクサは、WAIT_FOR_XICC_VECベクトル信号のビットの中でこのエントリに対応するビットを、WAIT_FOR_XICC信号として出力する。従って、この信号は、現在受入れられつつあるトラップのトラップレベルに現在マッピングされているトラップスタック820のエントリが、トラップが受入れられたときに論理XICCレ

ジスタとしてマッピングされた物理レジスタの内容を待っているか否か、を表す。

図45を参照すると、XICC捕捉ロジック823はISU 200に対してWAIT_FOR_XICC信号を出力する。もしこの信号が提示されれば、ISU 200は、この信号が提示されなくなるまで、RETRY命令またはDONE命令を発しない。

ISU 200が最後にRETRY命令またはDONE命令を発したとき、該ISU 200は、DONE命令が発せられたか否かを示す信号とRETRY命令が発せられたか否かを示す信号とを含むDONE_RETRY_IS信号を、出力する。

図47を参照すると、読込マルチプレクサ835はRD_ENTRY信号として、現在の

TL値に対応するRENAME_MAP信号を出力する。更に図45を参照すると、トラップスタックユニット815のRD/WR制御ロジック822は、DONE命令またはRETRY命令が既に発せられたことがDONE_RETRY_IS信号により示されたときRD_ENTRY信号により識別される記憶エレメントから、TPC、TNPC、ASI、CWP、PSTATEおよびXICCの各値を、読み出す。

図17を参照すると、ASI、CWPおよびPSTATEの各値は、DONE_RETRY_IS信号に応じ、制御レジスタファイル800のRD/WR/UPDATEロジック816により対応レジスタ内に書込まれる。

更に、図45に示される様に、TPCおよびTNPCの各値はRD_TPC__TNPC信号を以てBRB 59に付与される。その結果、BRB 59はこの値を使用してPCおよびNPCの各値を形成して、次の命令セットをフェッチする。

最後に、XICC値はRD_XICC信号を以てDFB 62に供与される。図8を参照すると、DONE_RETRY_IS信号に応じ、CCRFRN 610は論理XICCレジスタを物理レジスタにマッピングする。従って、CCRFRN 610は

トラップスタックユニット815からXICCデータを受けたとき、それを新たにマッピングされた物理レジスタ内に記憶する。

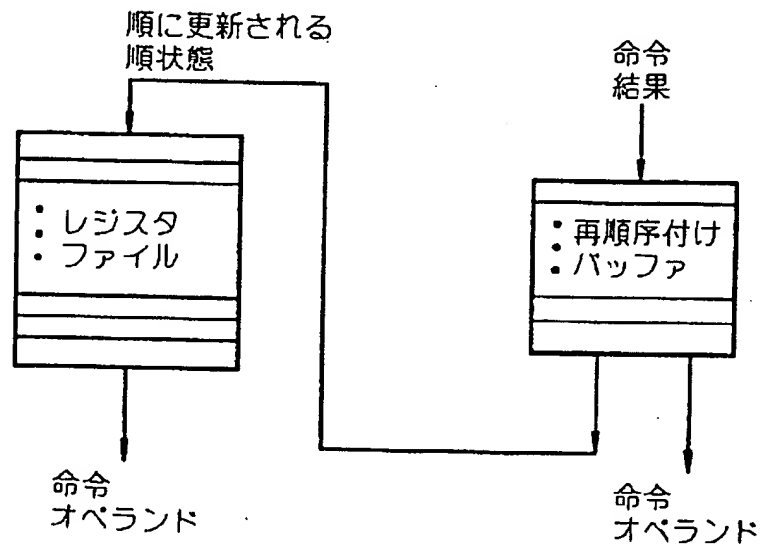
更に、当業者であれば、FXU 601およびFXAGU 602の各々が、論理XICCレジスタを変更する固定小数点命令を実行し得る1個以上の実行ユニットを有し得ることを理解し得よう。この場合、XICC捕捉ロジック823は、トラップが受入れられるときに論理XICCレジスタとしてマッピングされた物理レジスタの内容を以てトラップスタック820を適切に更新すべく既述したロジック、の複製ロジックを有することになる。

これに加え、当業者であれば、XICC捕捉ロジック823の概念を他の型式のレジスタまで拡張し得ることを理解し得ようが、斯かる他の型式のレジスタとは、その内容が、トラップが受入れられた時点では利用する必要は無いがトラップスタックには書込まれる必要があるレジスタ、および、レジスタ名称変更（即ち、論理レジスタに対する物理レジスタのマッピング）が既に行われたレジスタ等である。

本明細書にて言及した公報および特許出願の全てを参照により援用するが、これは、個々の公報もしくは特許出願を特定のにかつ個別的に参照により援用したのと同じの程度まで援用する。

本発明を十分に説明してきたが、当業者であれば、添付の請求の範囲の精神および範囲から逸脱すること無しに多くの変更および修正を為し得ることは理解し得よう。

【図1】

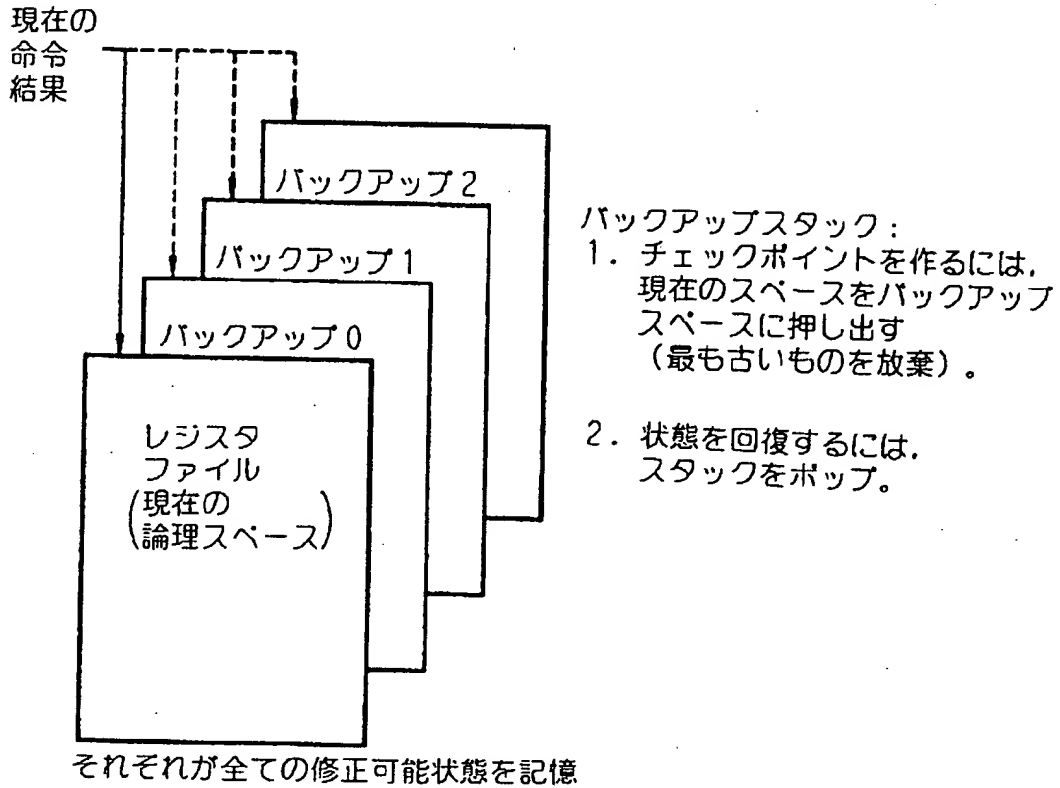


再順序付けバッファ構造

FIG. 1
(先行技術)

【図2】

チェックポイントの時点で利用不可能な結果は、適切なバックアップスペースに書き込まれ、順状態を完了



状態のチェックポイント記憶と復旧

FIG. 2
(先行技術)

【図3】

従来のチェックポイントレジスタでのチェックポイント状態

命令Aによって修正(mod) されうる状態:

mod	-	-	mod	-	-	-	-	-	-
-----	---	---	-----	---	---	---	---	---	---

命令Bによって修正(mod) されうる状態:

mod	-	-	-	-	-	-	-	-	-
-----	---	---	---	---	---	---	---	---	---

命令Cによって修正(mod) されうる状態:

-	mod	mod	-	mod	mod	mod	-	mod	mod
---	-----	-----	---	-----	-----	-----	---	-----	-----

命令Dによって修正(mod) されうる状態:

-	-	-	-	-	-	-	mod	-	-
---	---	---	---	---	---	---	-----	---	---

チェックポイントされた状態は、命令セットにおける命令のいずれか1つによって修正されうる全ての状態からなる:

状態1	状態2	状態3	状態4	状態5	状態6	状態7	状態8	状態9	状態10
-----	-----	-----	-----	-----	-----	-----	-----	-----	------

従来のチェックポイントレジスタは、実行可能な命令の内のいずれか1つによって修正されうる全ての状態を記憶する。

FIG. 3

【図4】

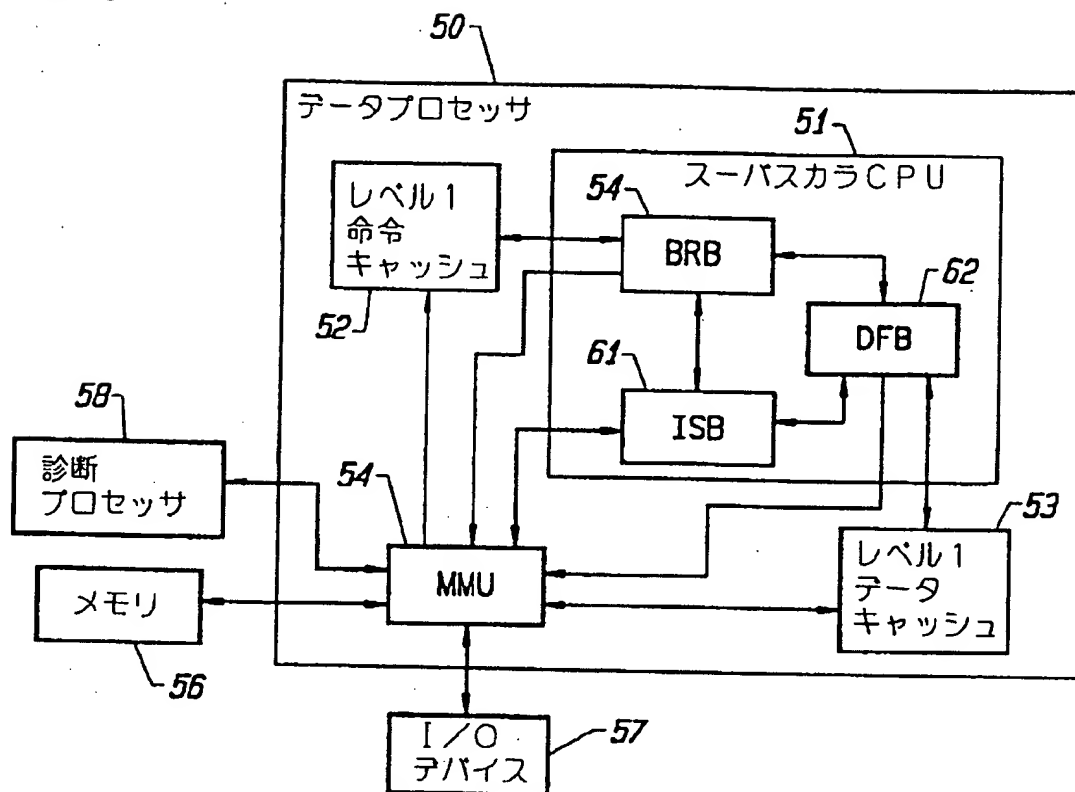


FIG. 4

【図5】

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
プログラム制御	フェッチ	発行 実行 完了	非起動	コミット	リタイア				
固定小数点 & 浮動小数点命令	フェッチ	発行	実行	完了	非起動	コミット	リタイア		
ロード/ストア命令 (キャッシュヒット)	フェッチ	発行	アドレス生成	キャッシュアクセス	データリターン	完了	非起動	コミット	リタイア
				実行					

FIG. 5

【図6】

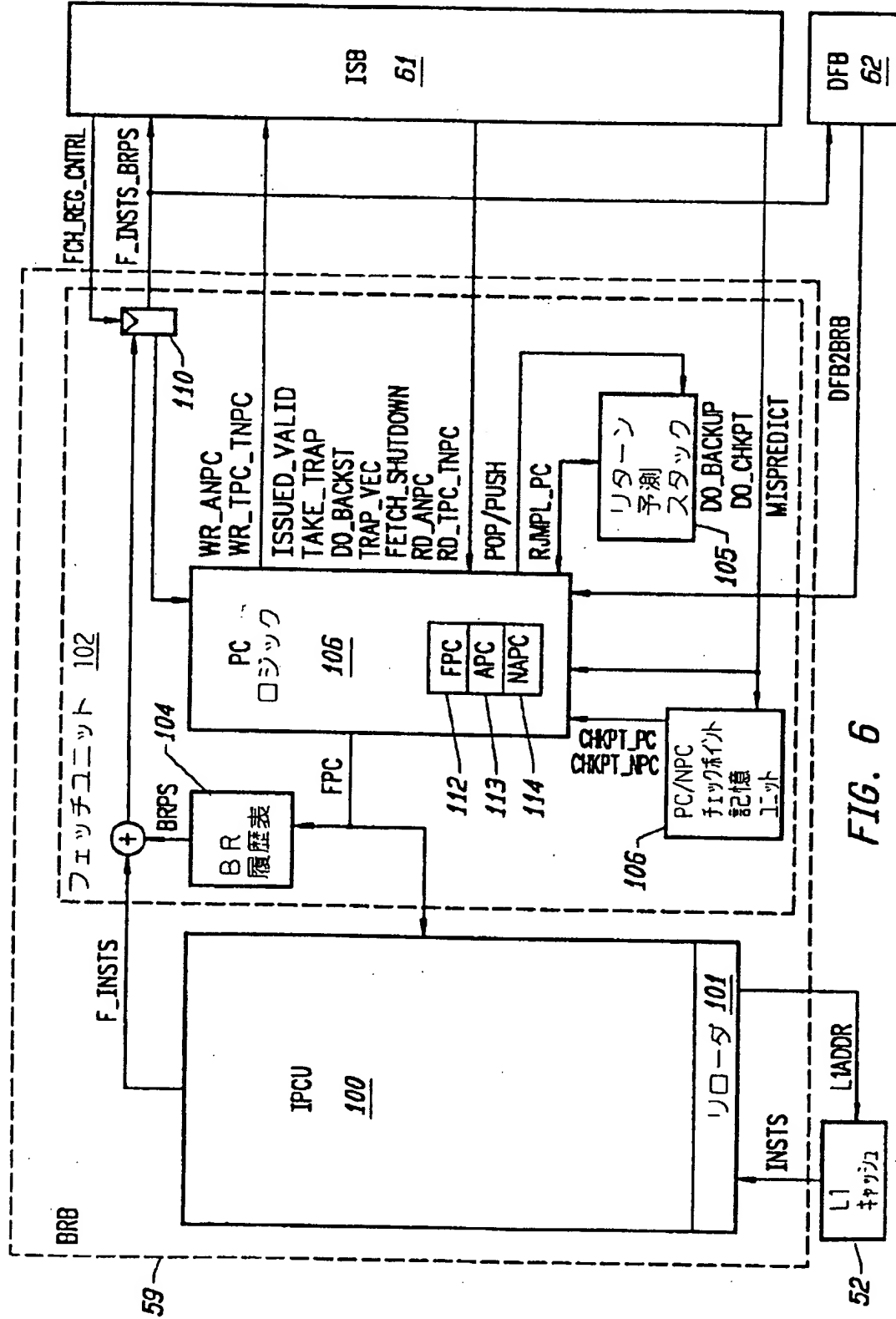


FIG. 6

【図7】

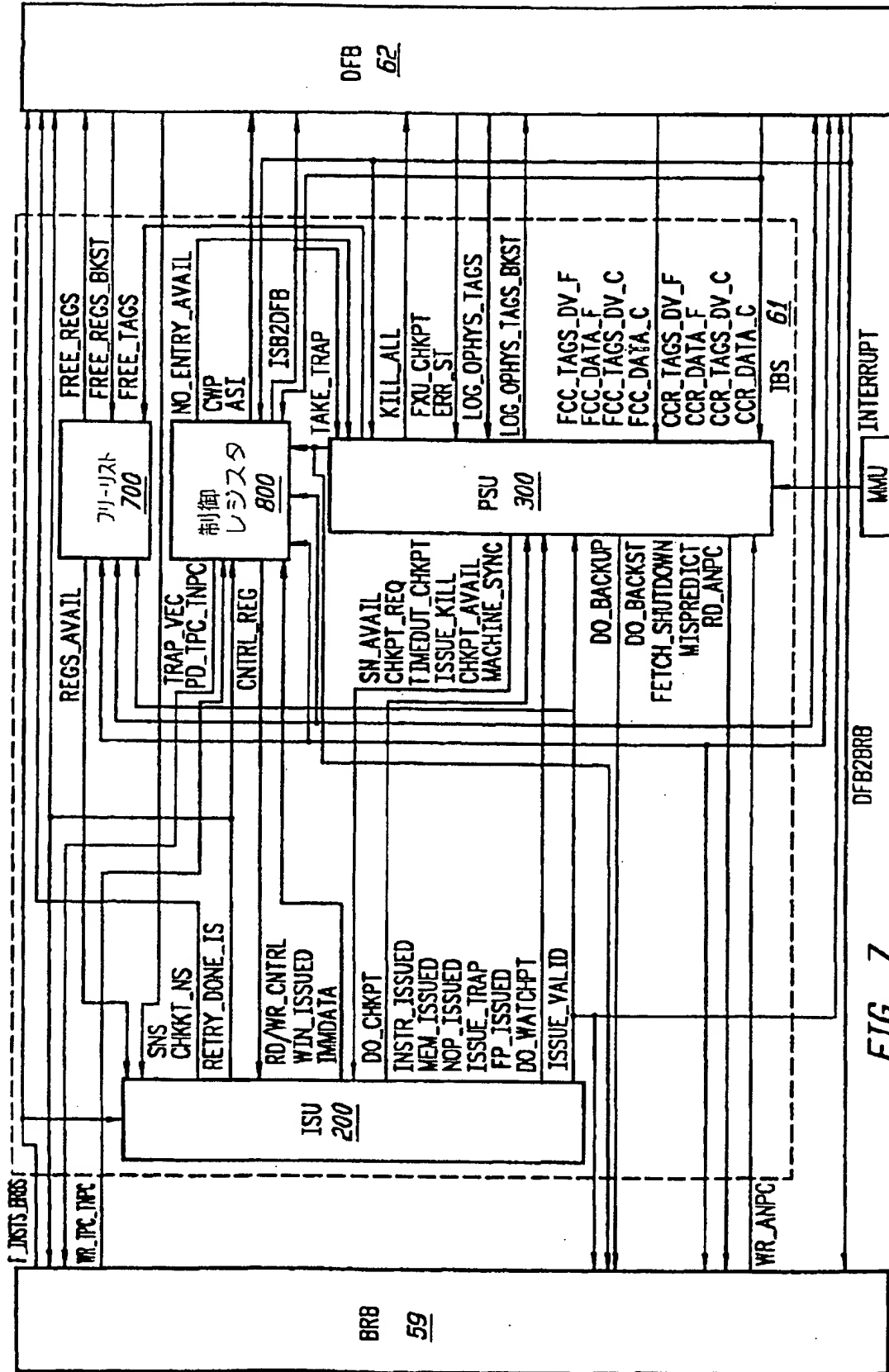


FIG. 7

【図8】

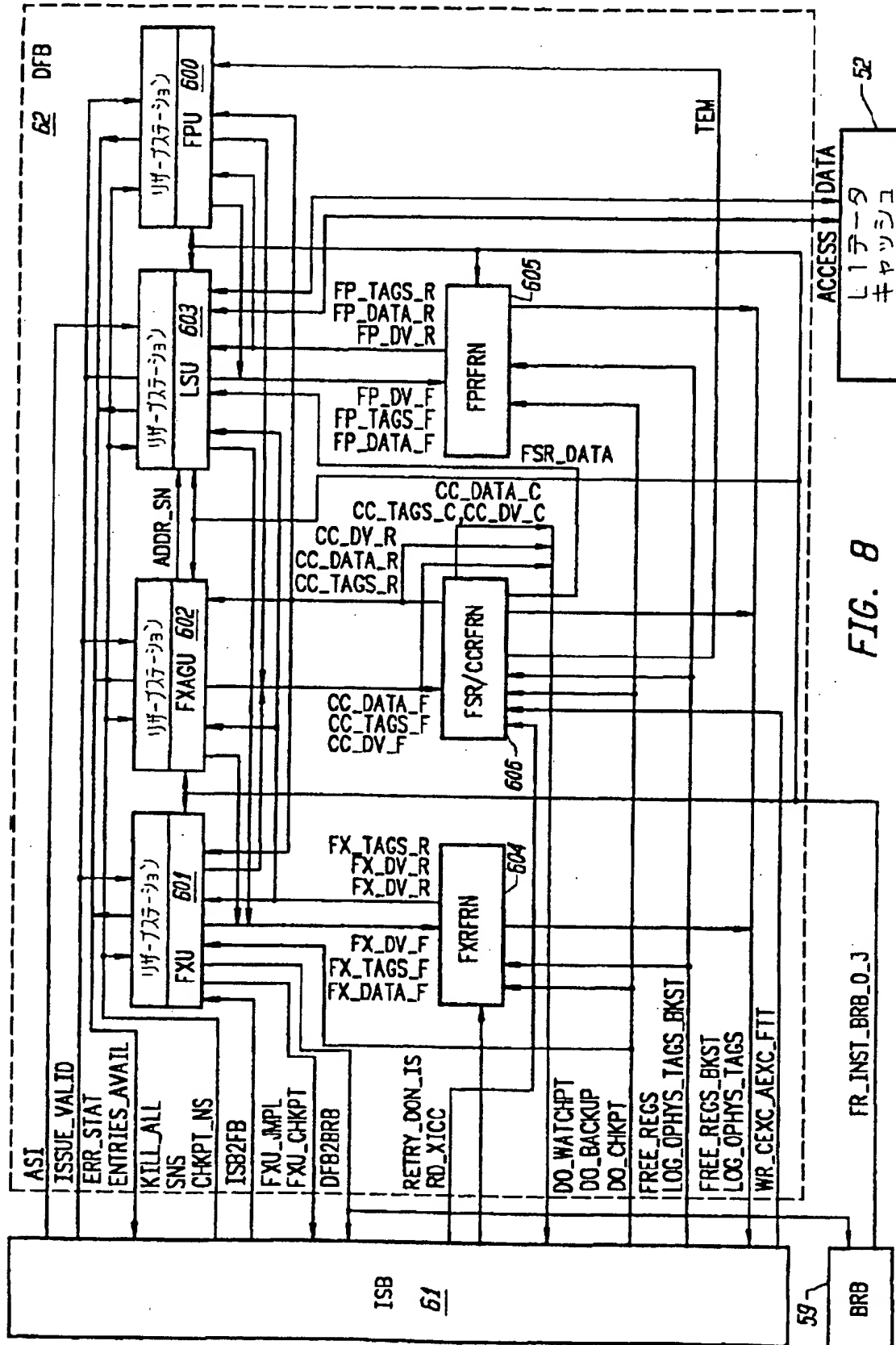


FIG. 8

【図9】

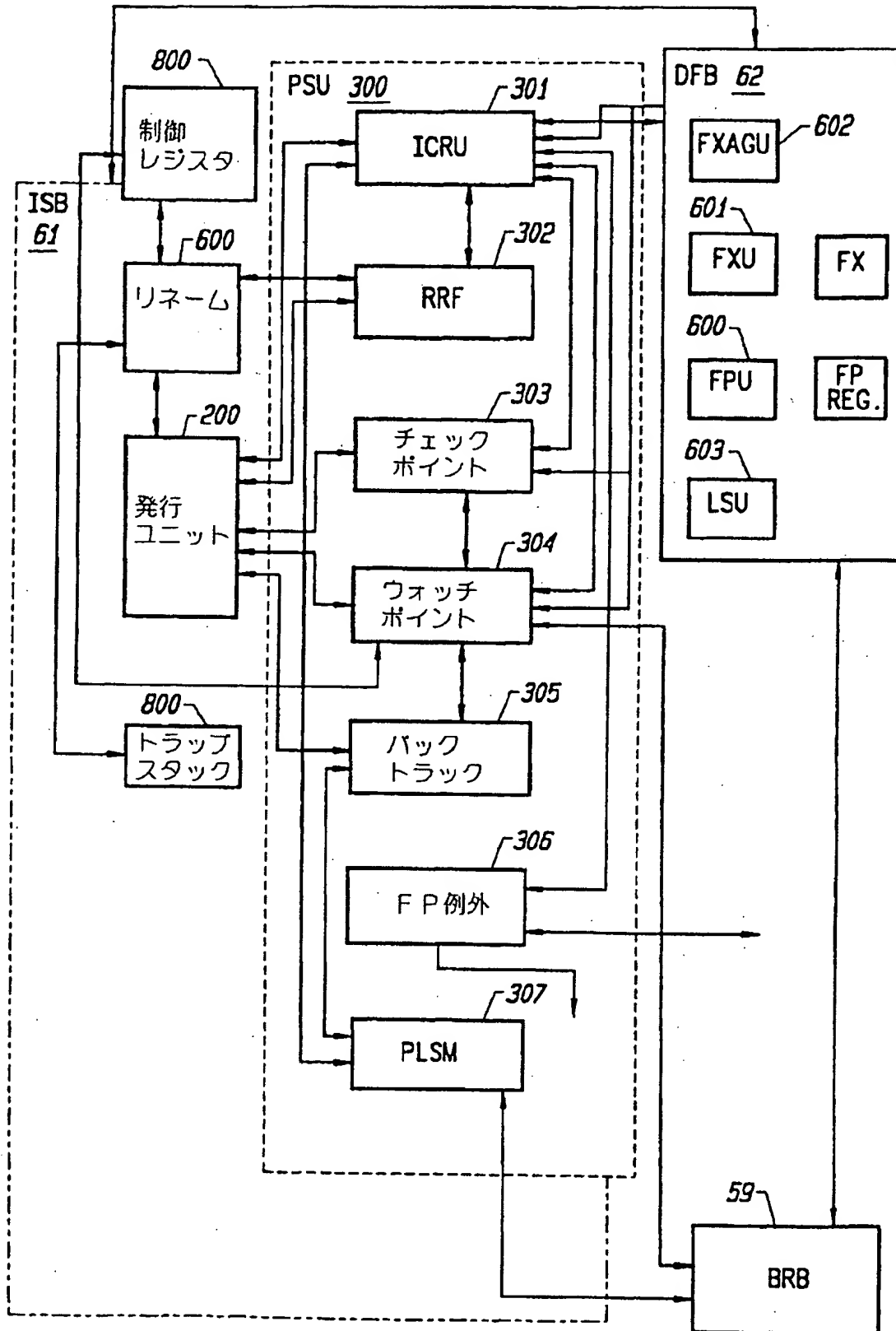


FIG. 9

【図10】

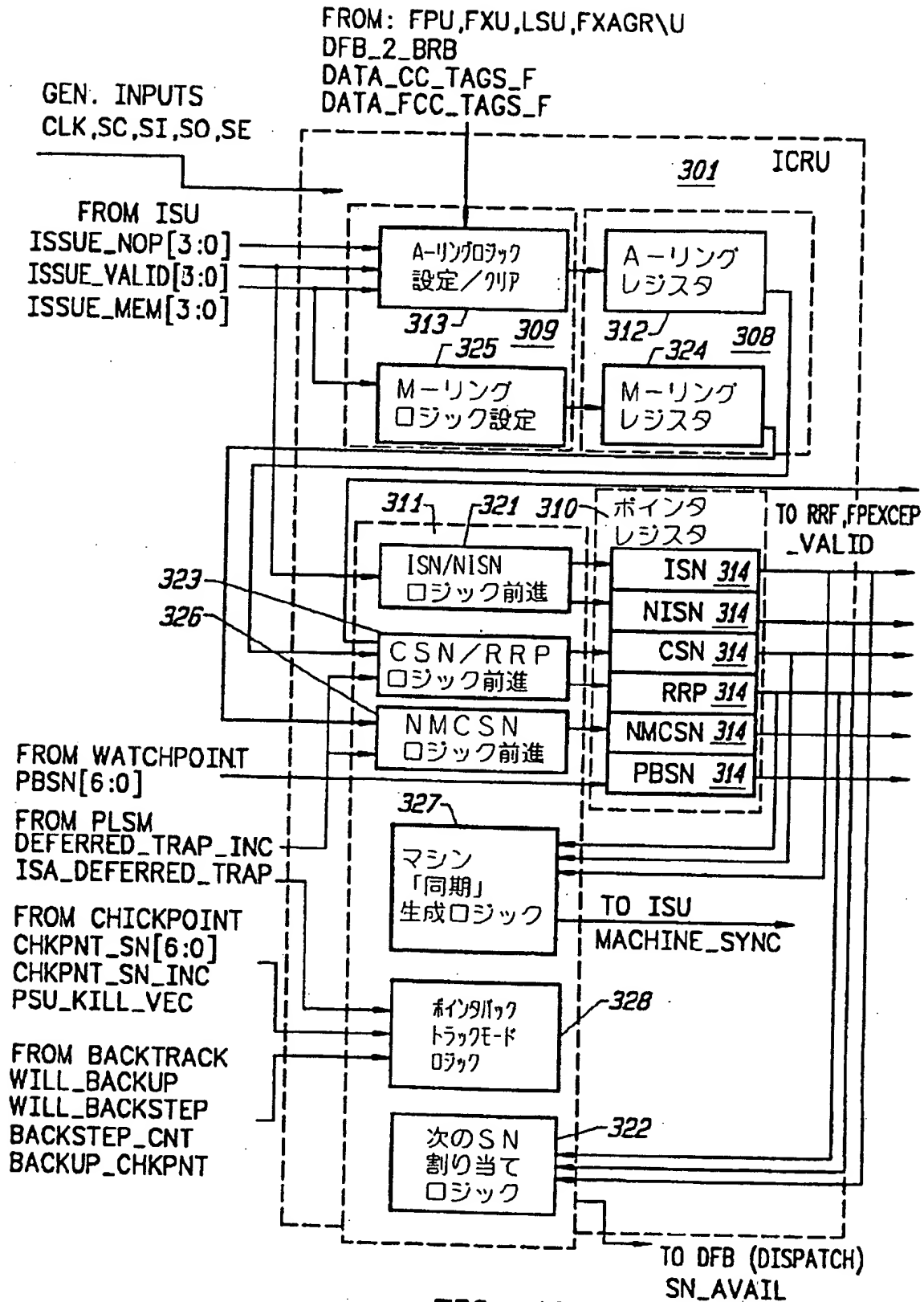


FIG. 10

【図11】

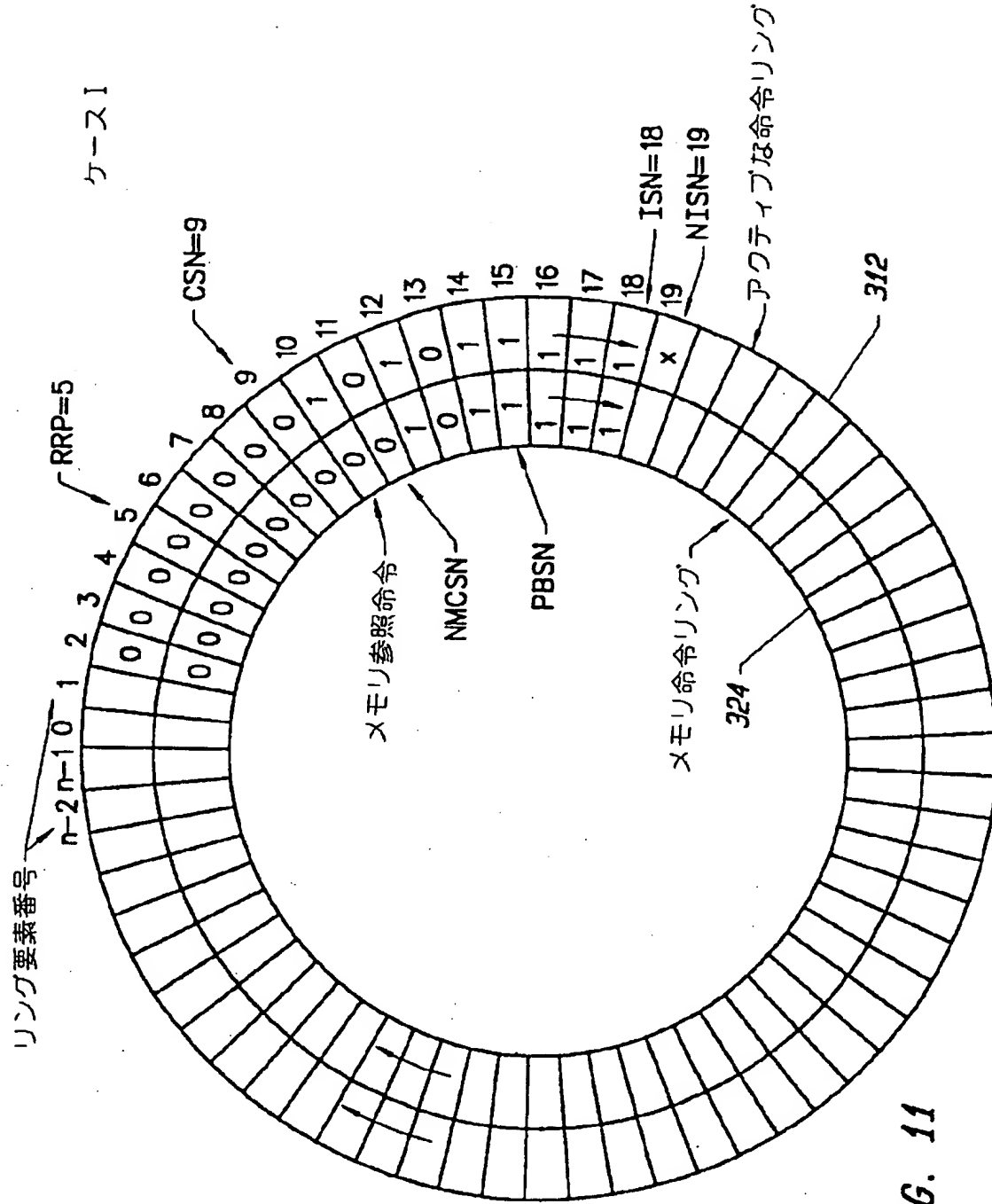
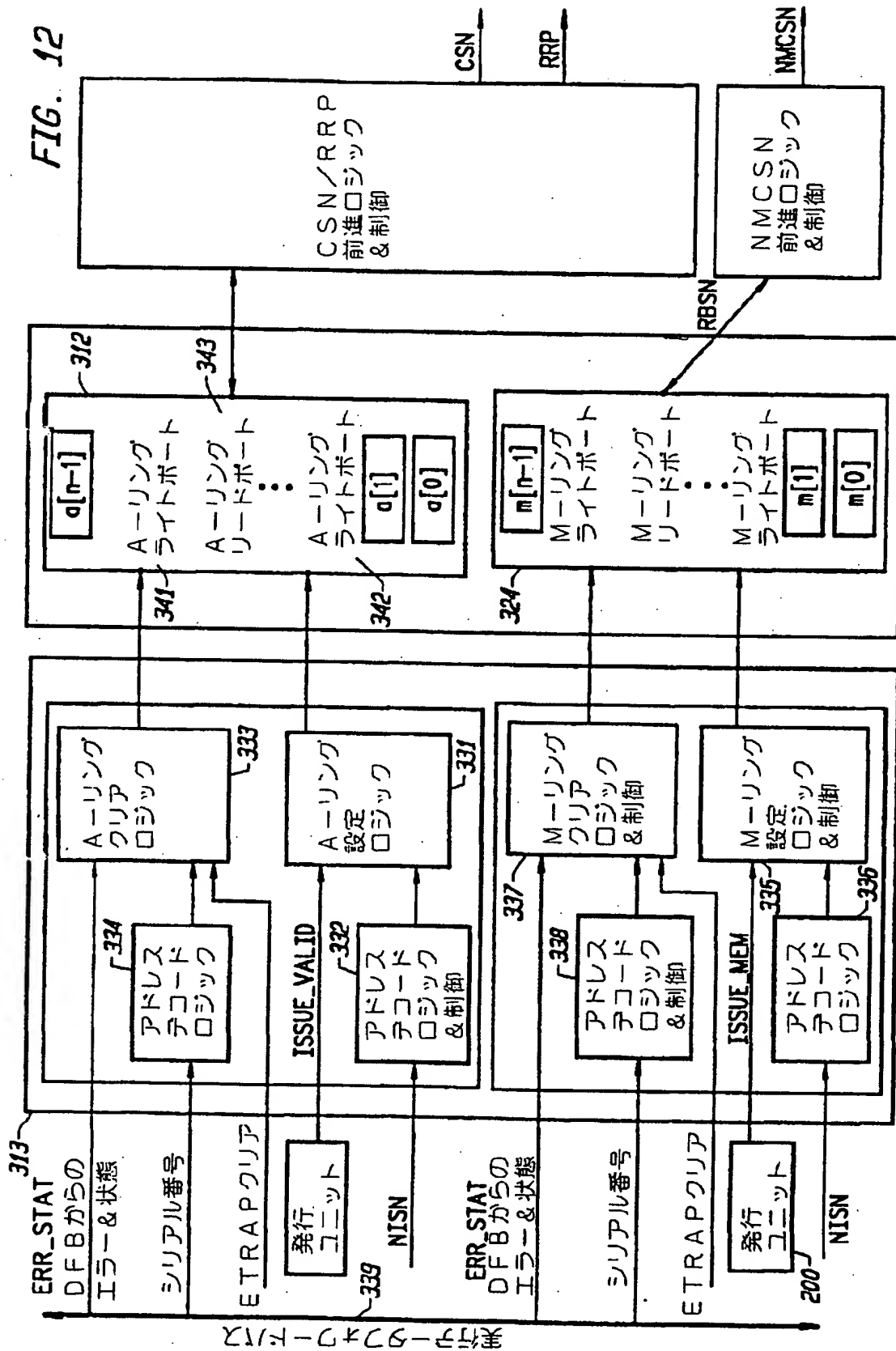


FIG. 11

【図12】



【図13】

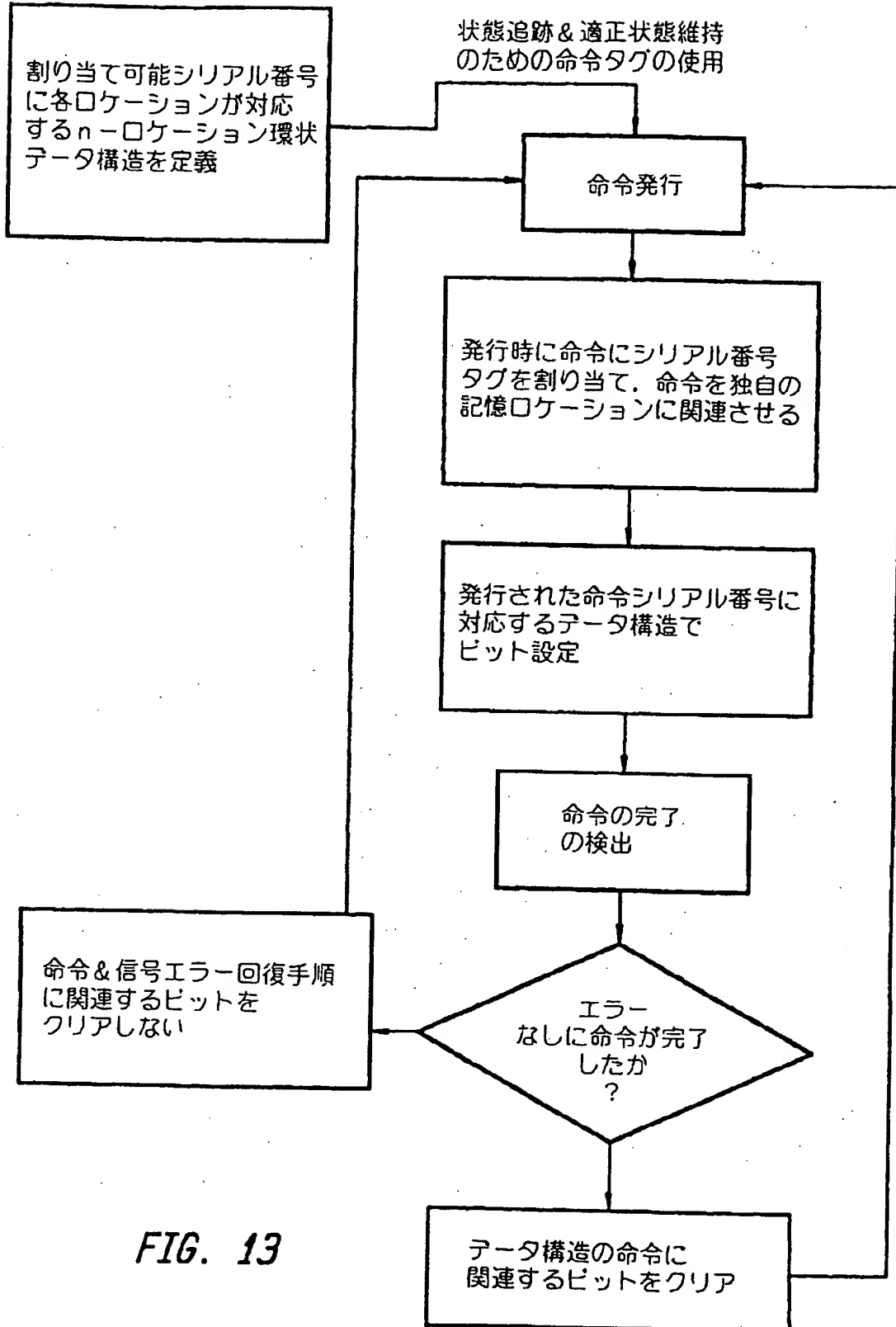
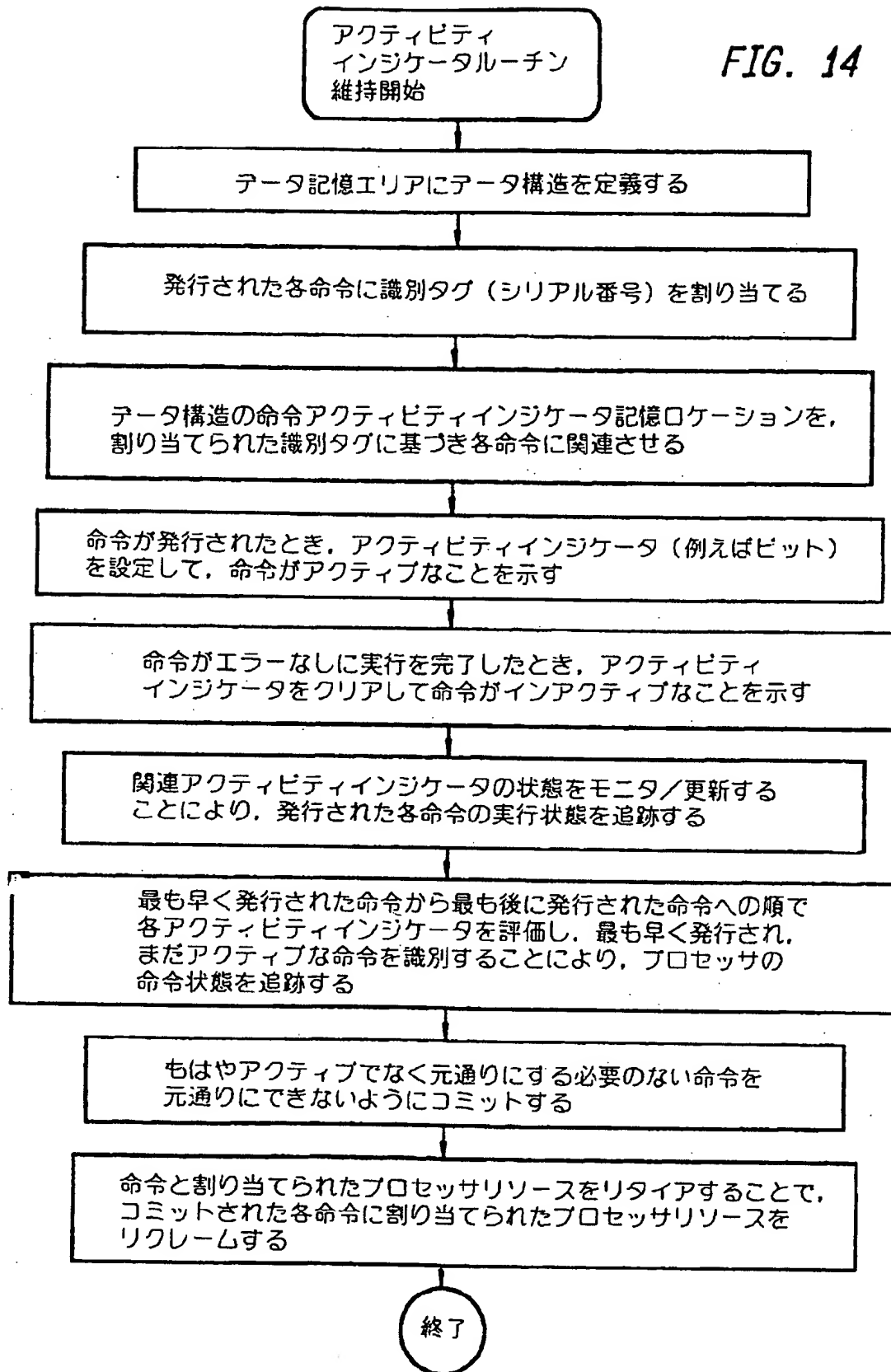


FIG. 13

【図14】

FIG. 14



【図15】

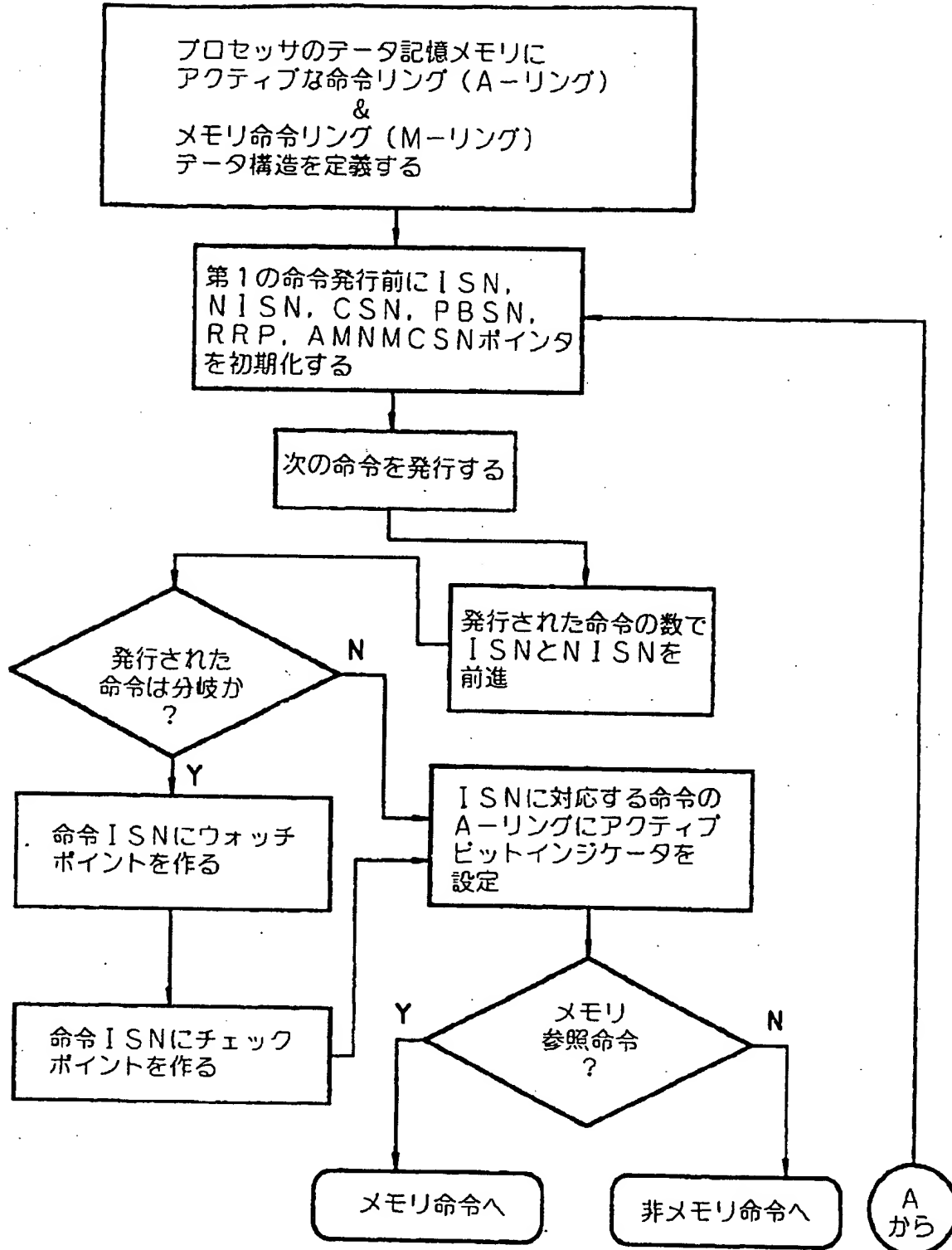


FIG. 15A

【図15】

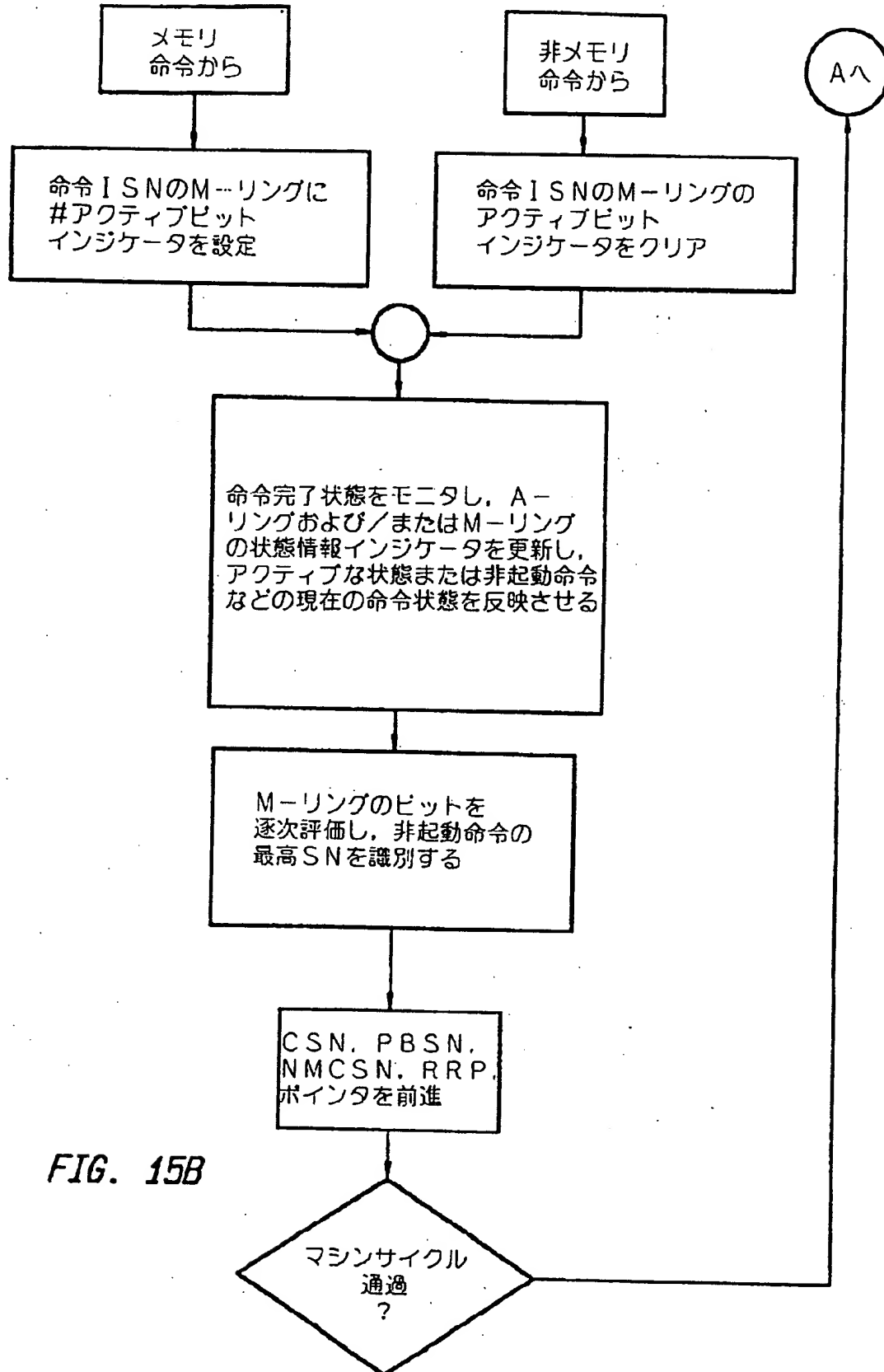


FIG. 15B

【図17】

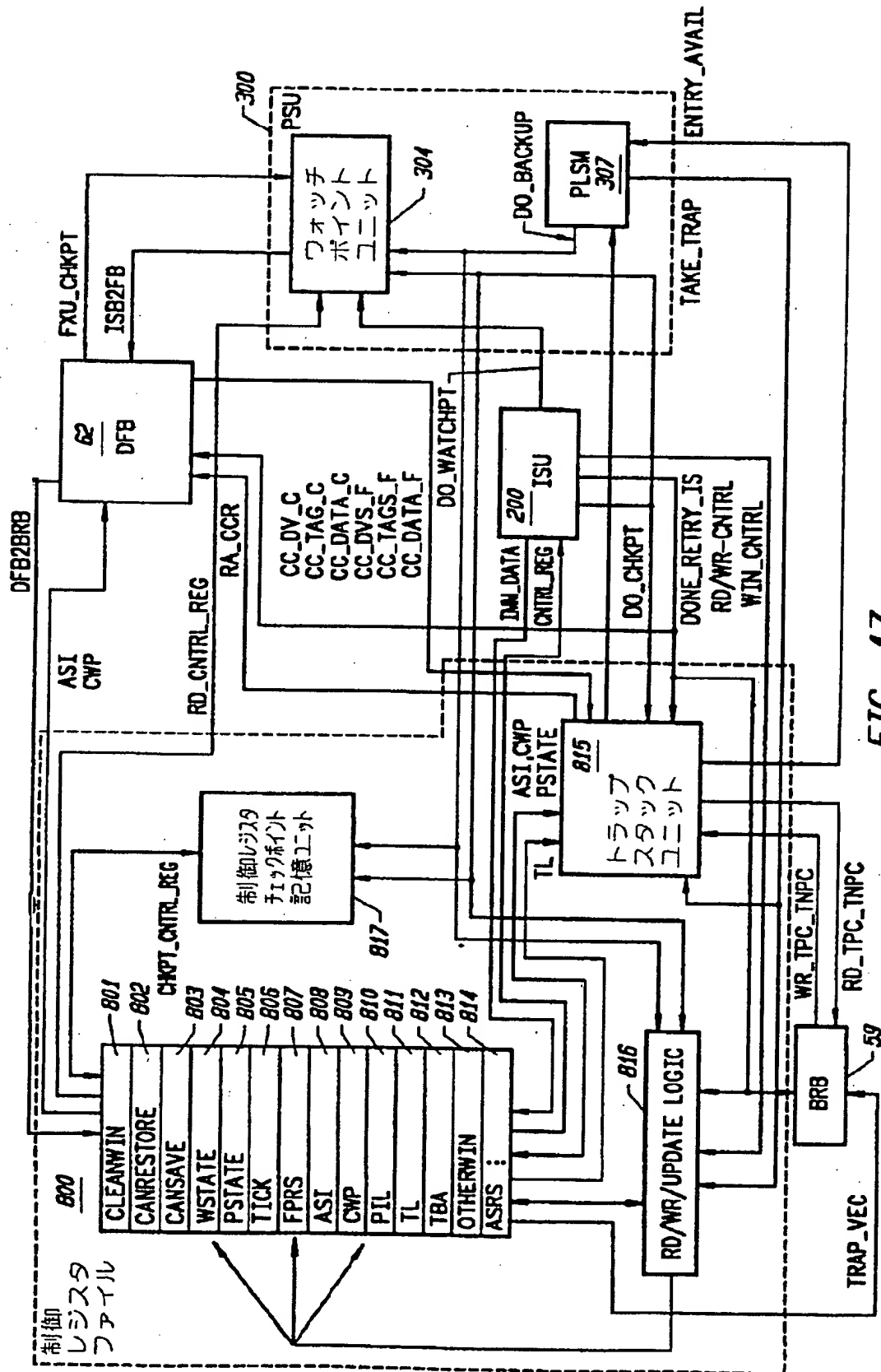


FIG. 17

【図18】

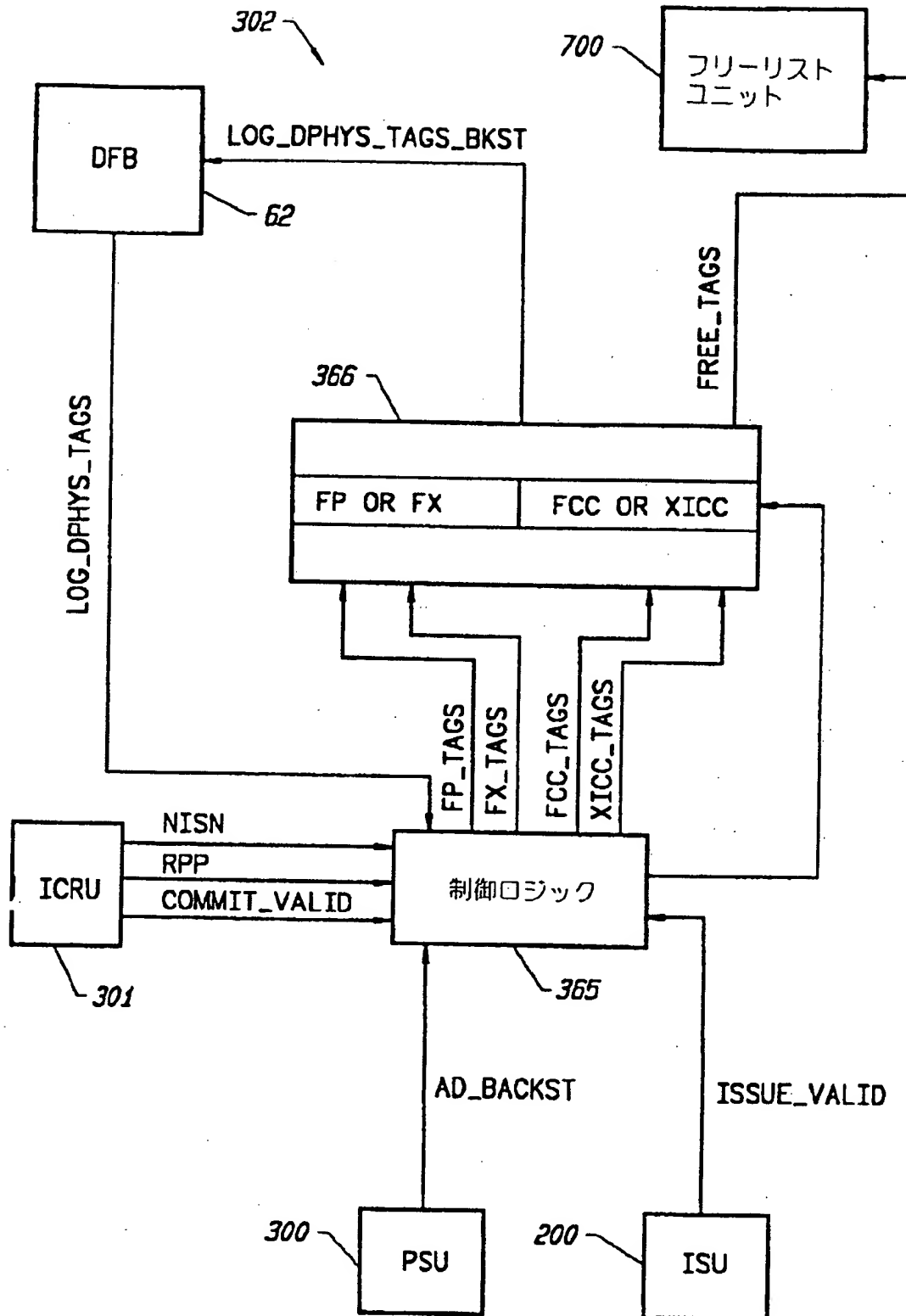
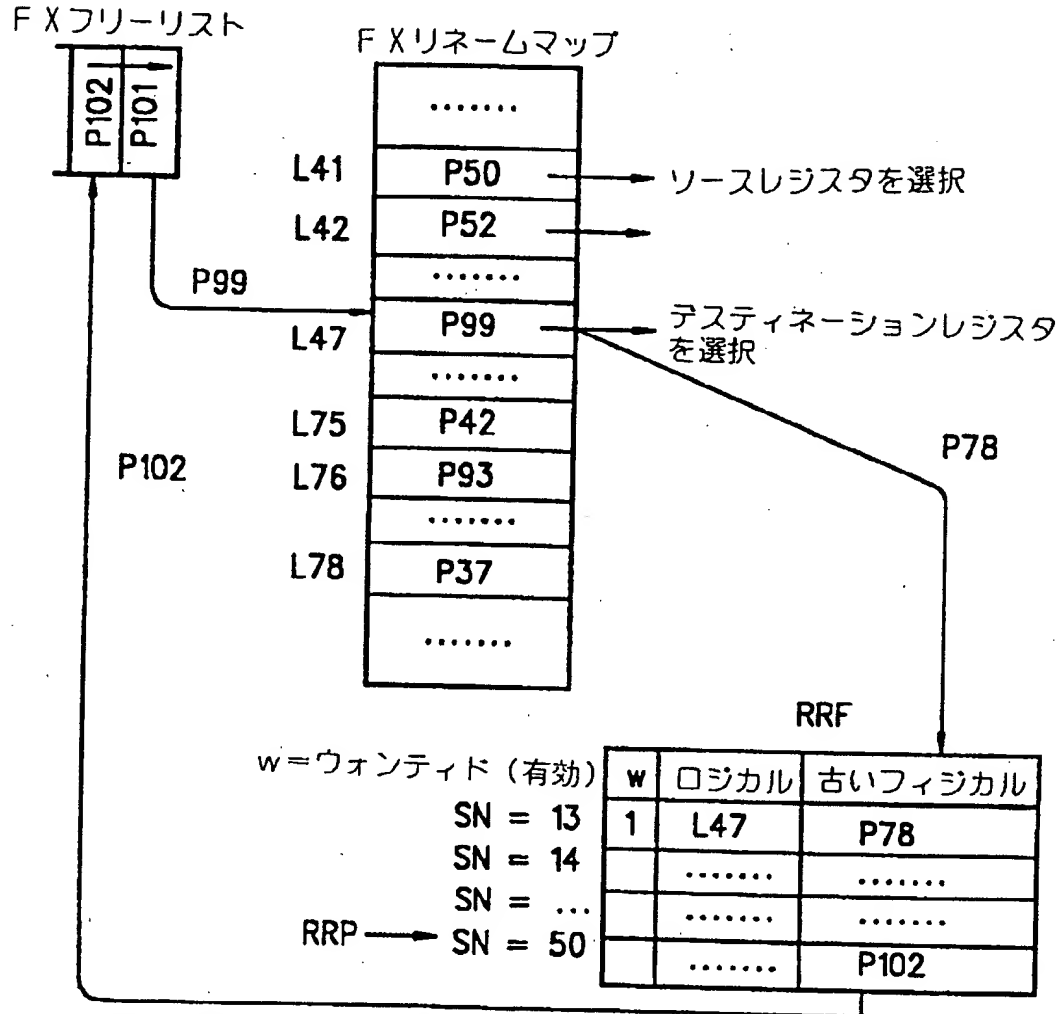


FIG. 18

【図19】

リネーミング例

- ・次のアドレスを実行したとき何が起きるかを見てみる：
 - アドレス L41, L42, L47 ! 想定 SN=13



- ◇ L41およびL42はP50およびP52にマッピングされたソースのフェッチに用いられる。
- ◇ フリーリストからレジスタP99を使用してテストレジスタL47をリネームした。
 ⇒ そのためP101はフリーリストの前に移動した。
- ◇ L47 (P78) の古い値は現在のSN (13) で索引付けされたRRFエントリに移動した。
 ⇒ P78が前回マッピングしたロジカルレジスタもRRFに記憶し、ウォンティド(w)ビットを設定してエントリが有効であることを示す。
- ◇ RRP=50, ゆえにP102はもはや必要なく、フリーリストに移動可能である。

FIG. 19

【図20】

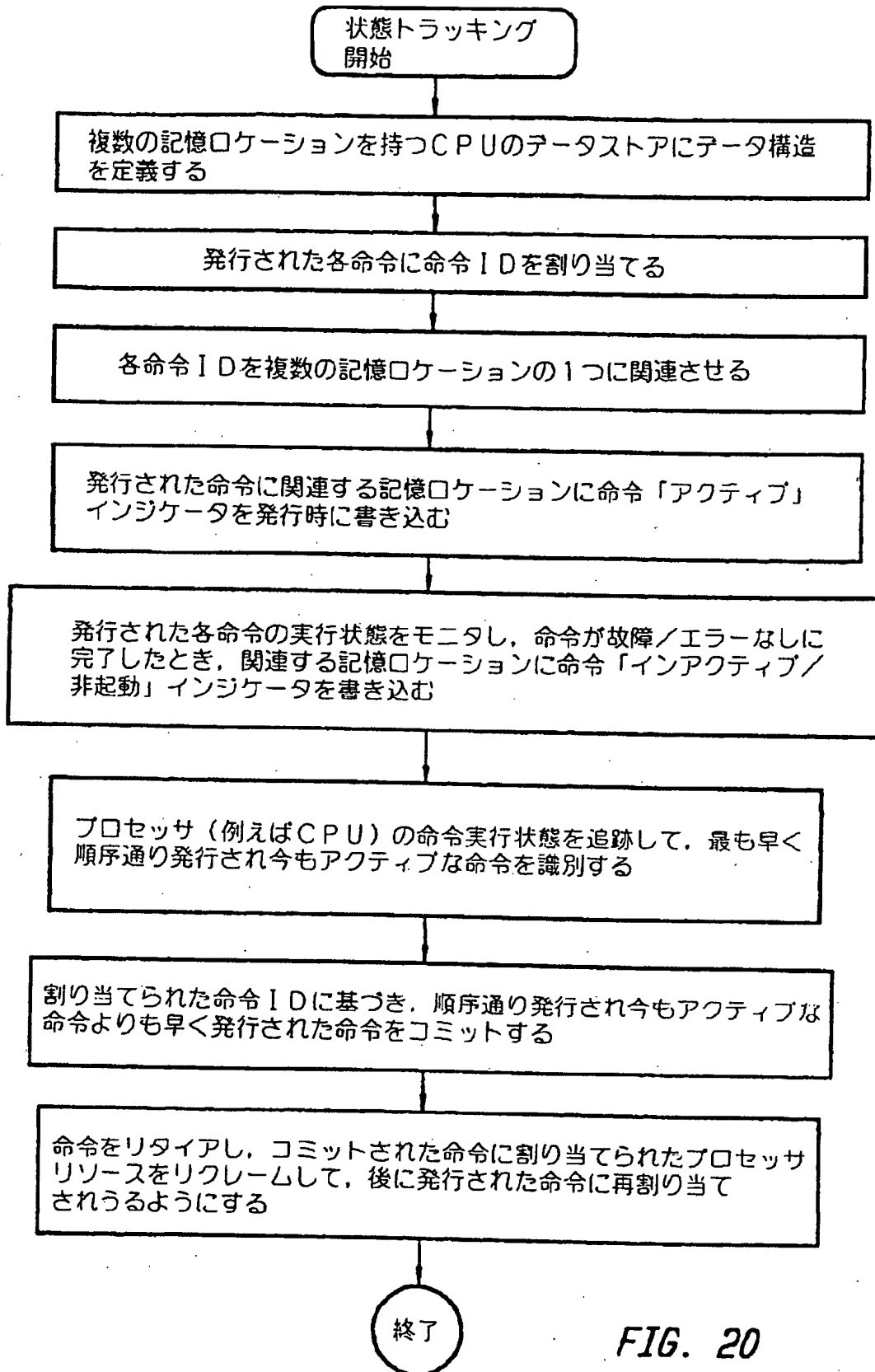


FIG. 20

【図21】

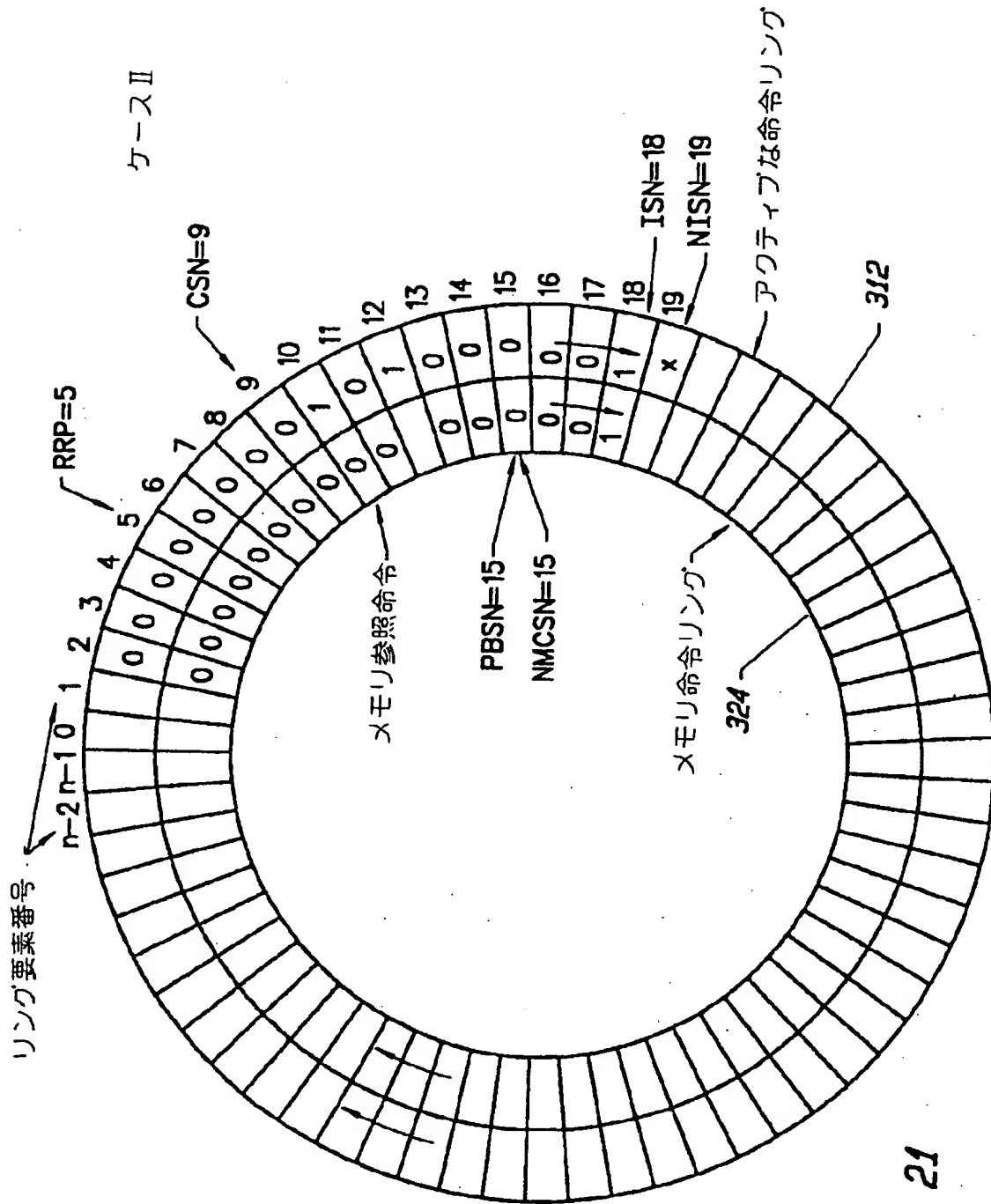


FIG. 21

【図23】

ロード/ストア命令の積極的スケジューリング

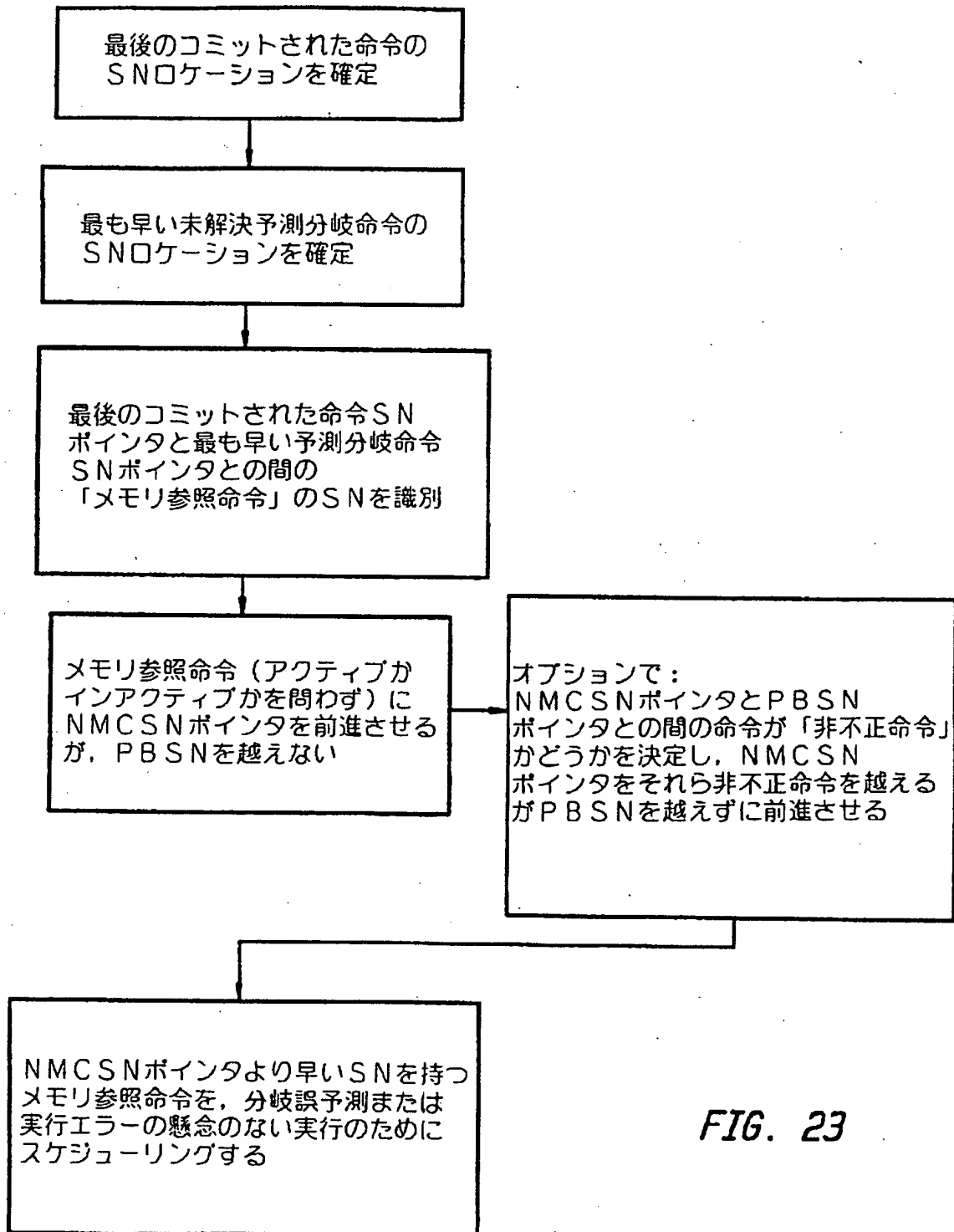


FIG. 23

【図24】

一般の

CLK, SC, SI, SE, SO

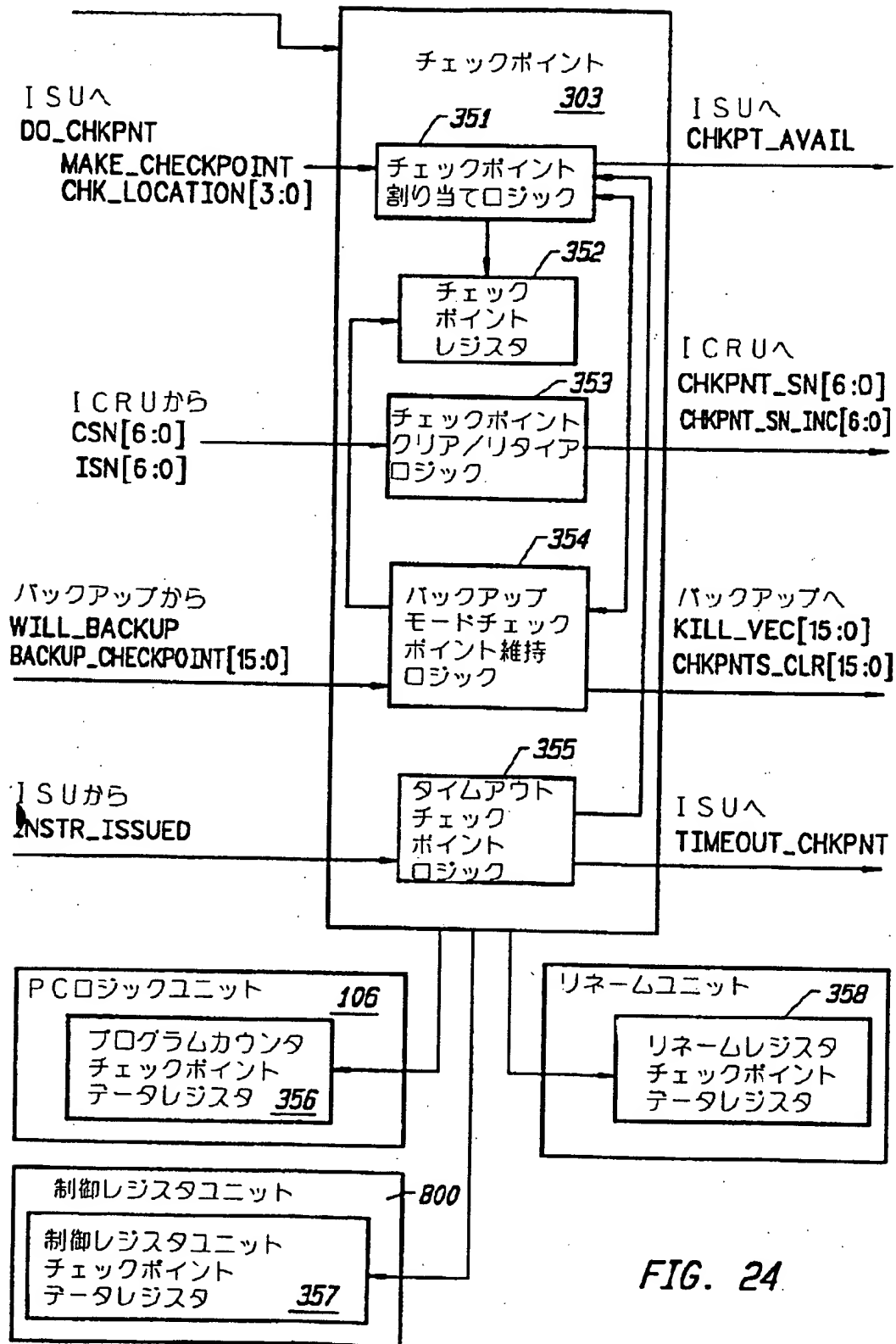


FIG. 24

【図25】

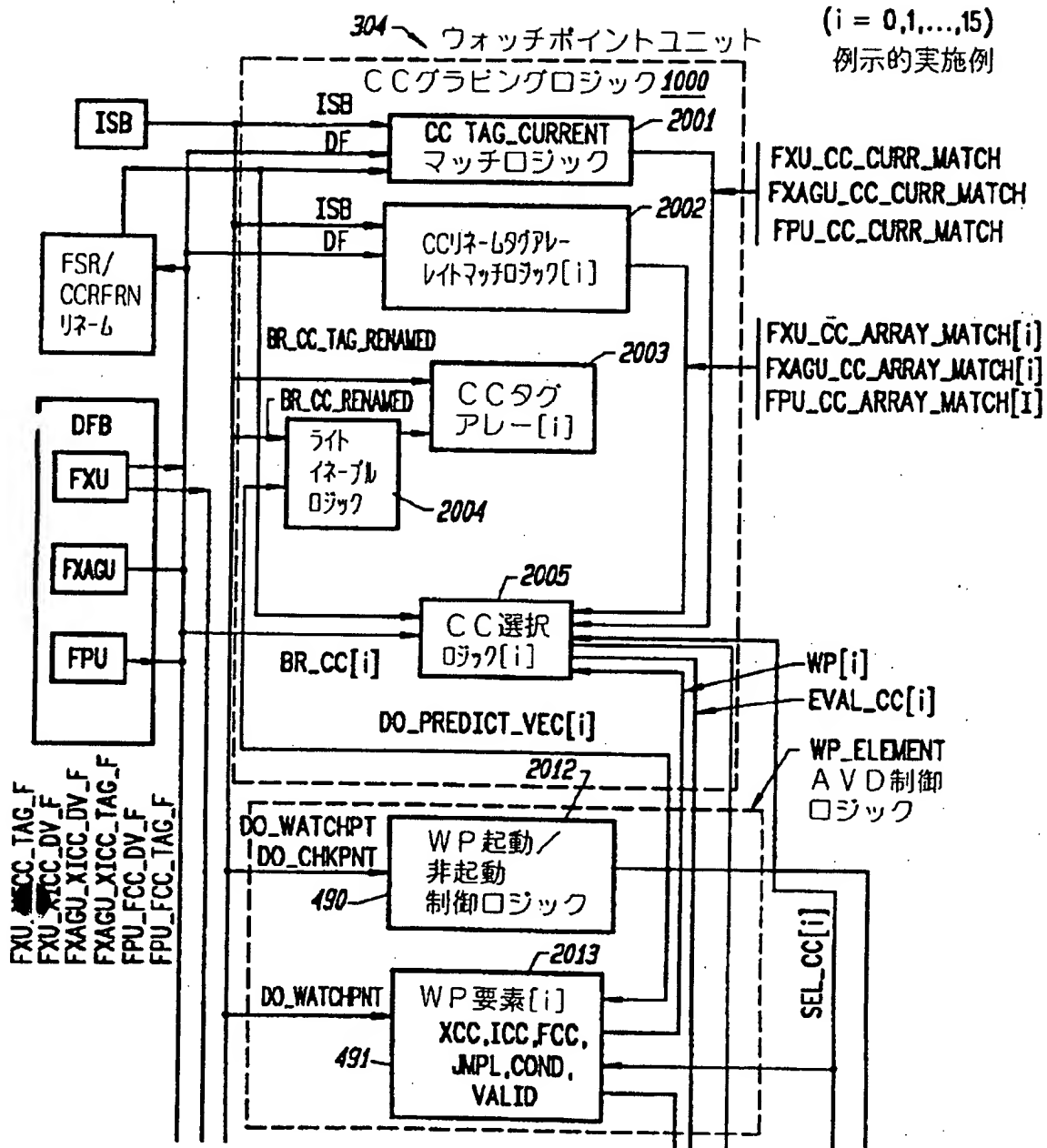


FIG. 25B

FIG. 25A

【図 25】

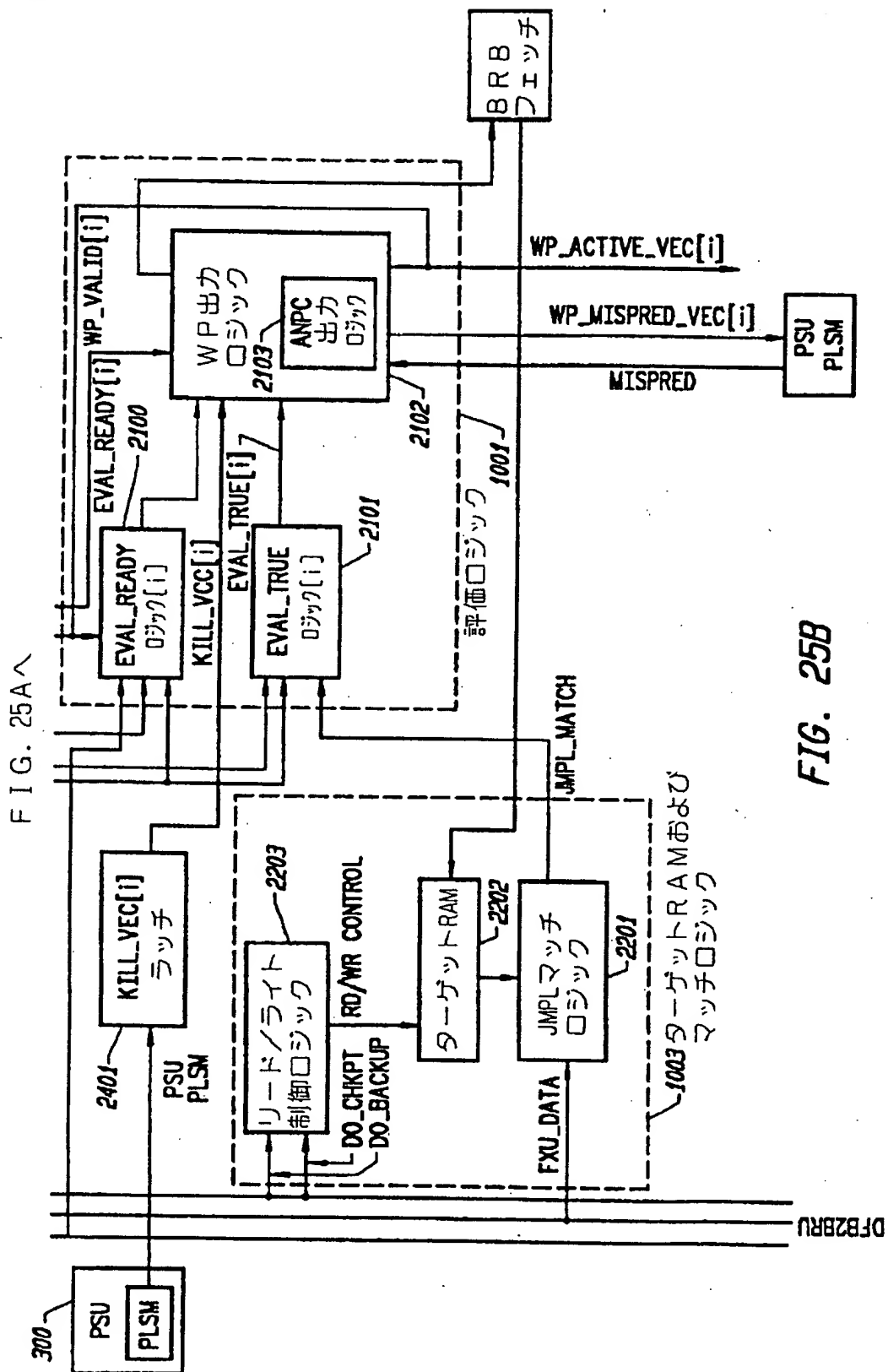


FIG. 25B

【図26】

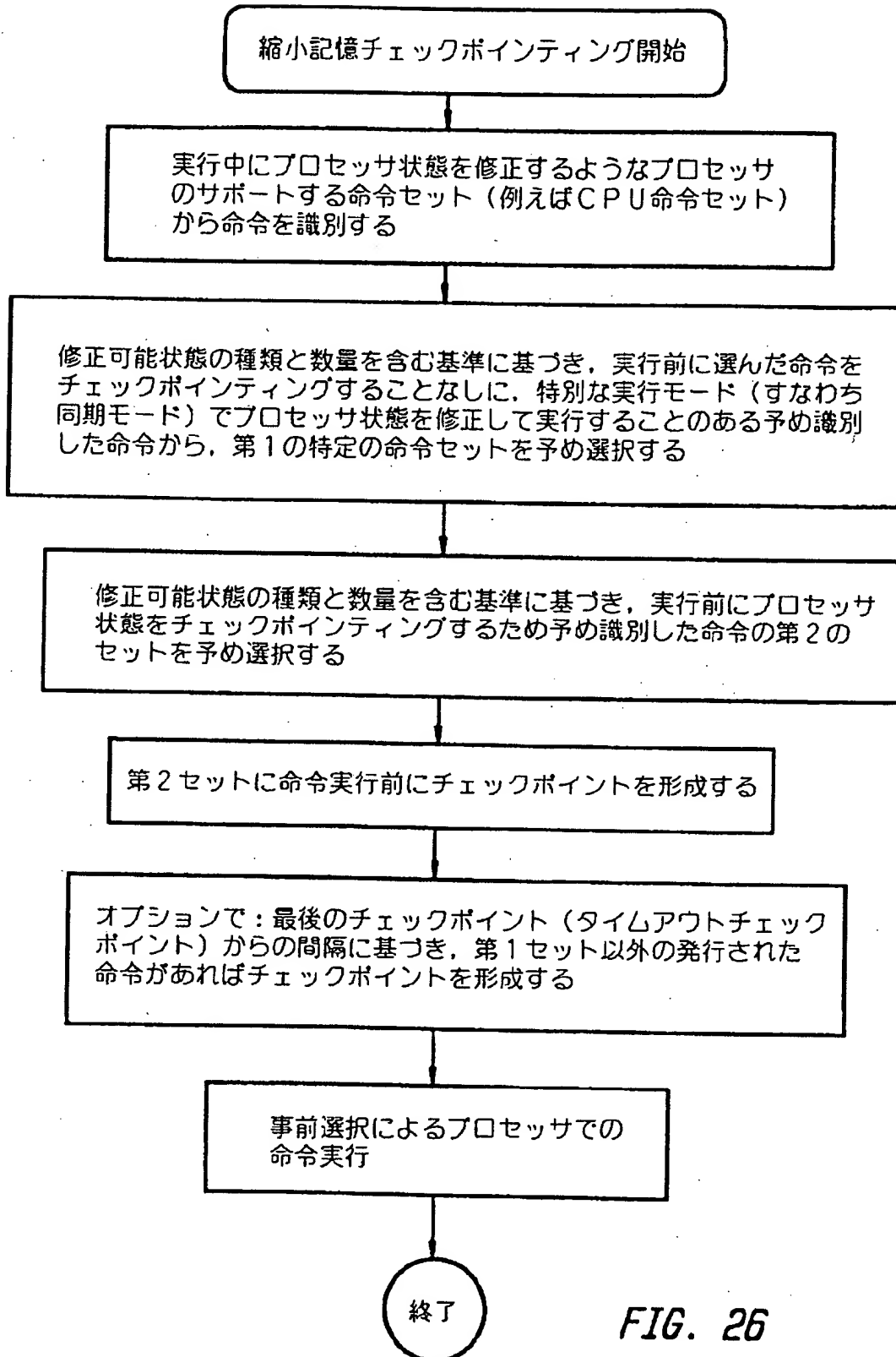
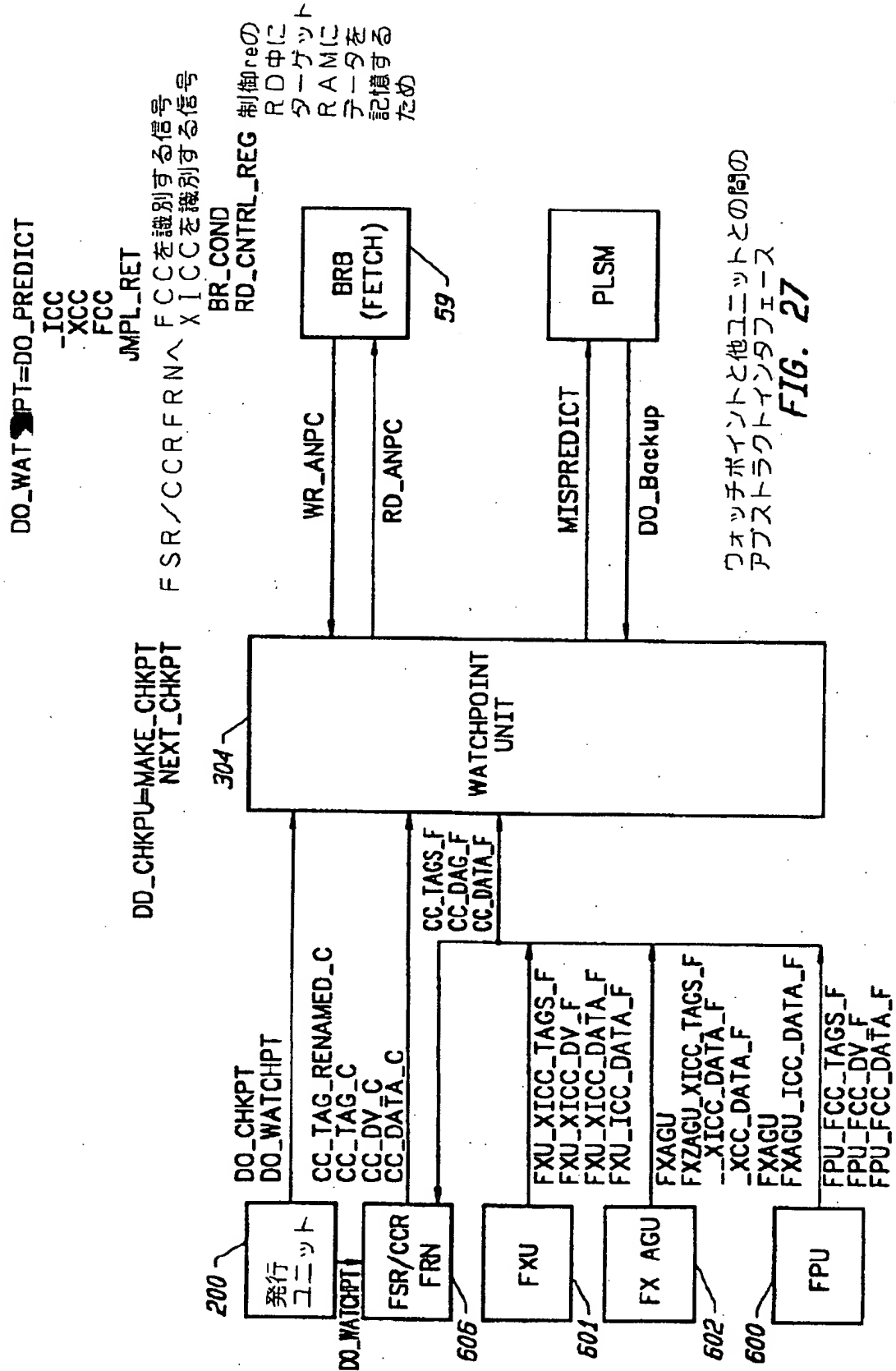


FIG. 26

【図27】



【図28】

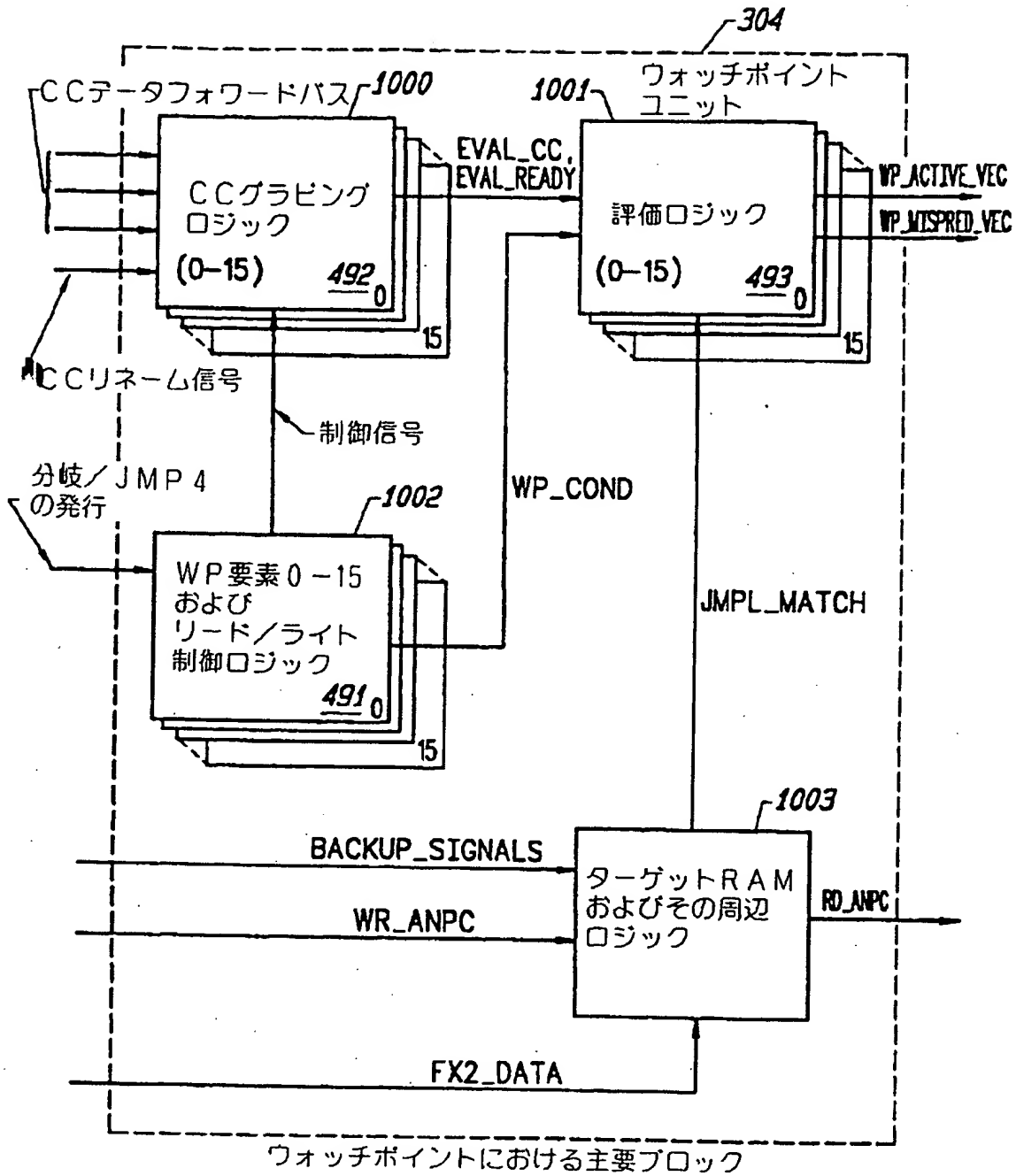


FIG. 28

【図29】

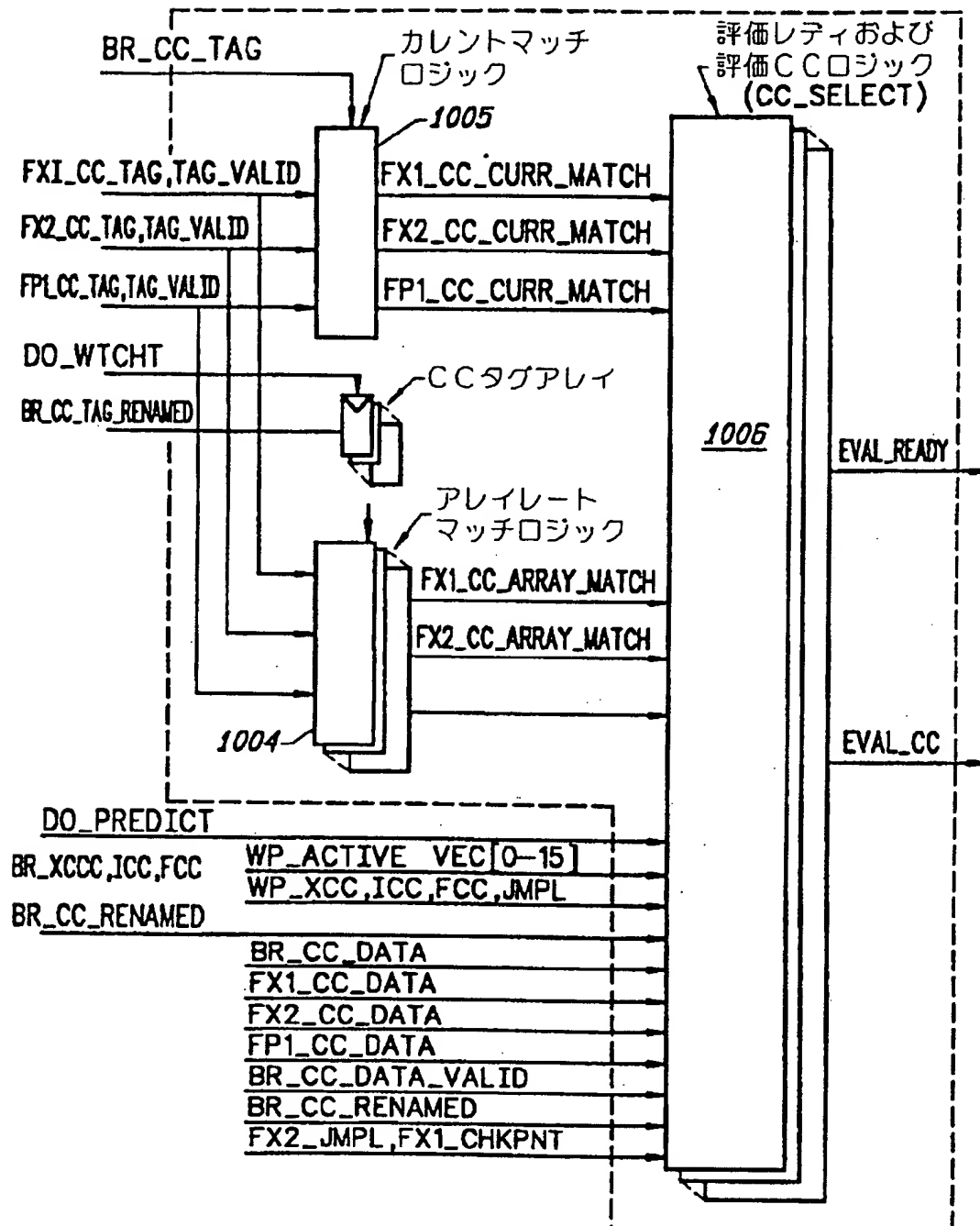
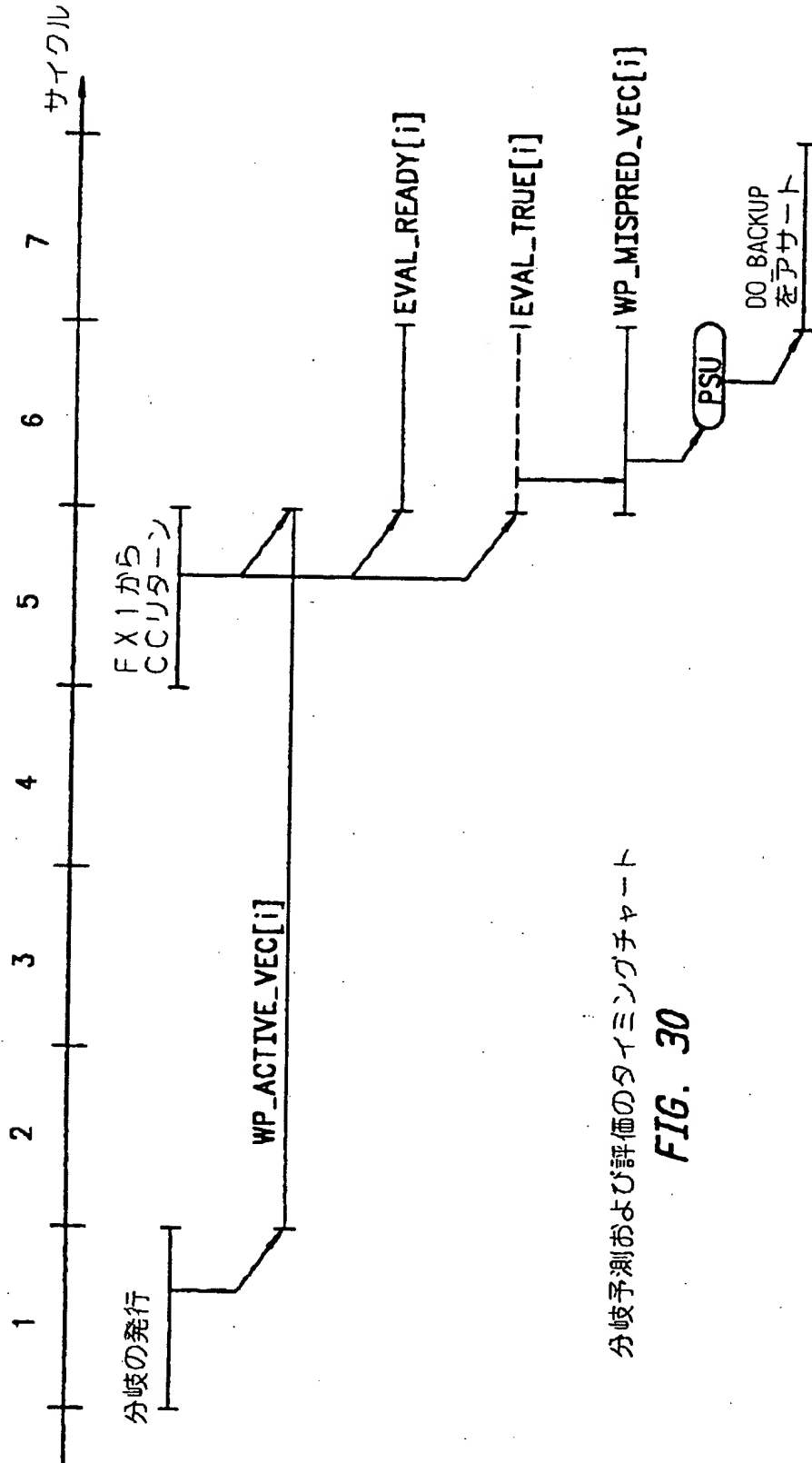


FIG. 29

【図30】



分岐予測および評価のタイミングチャート

FIG. 30

【図31】

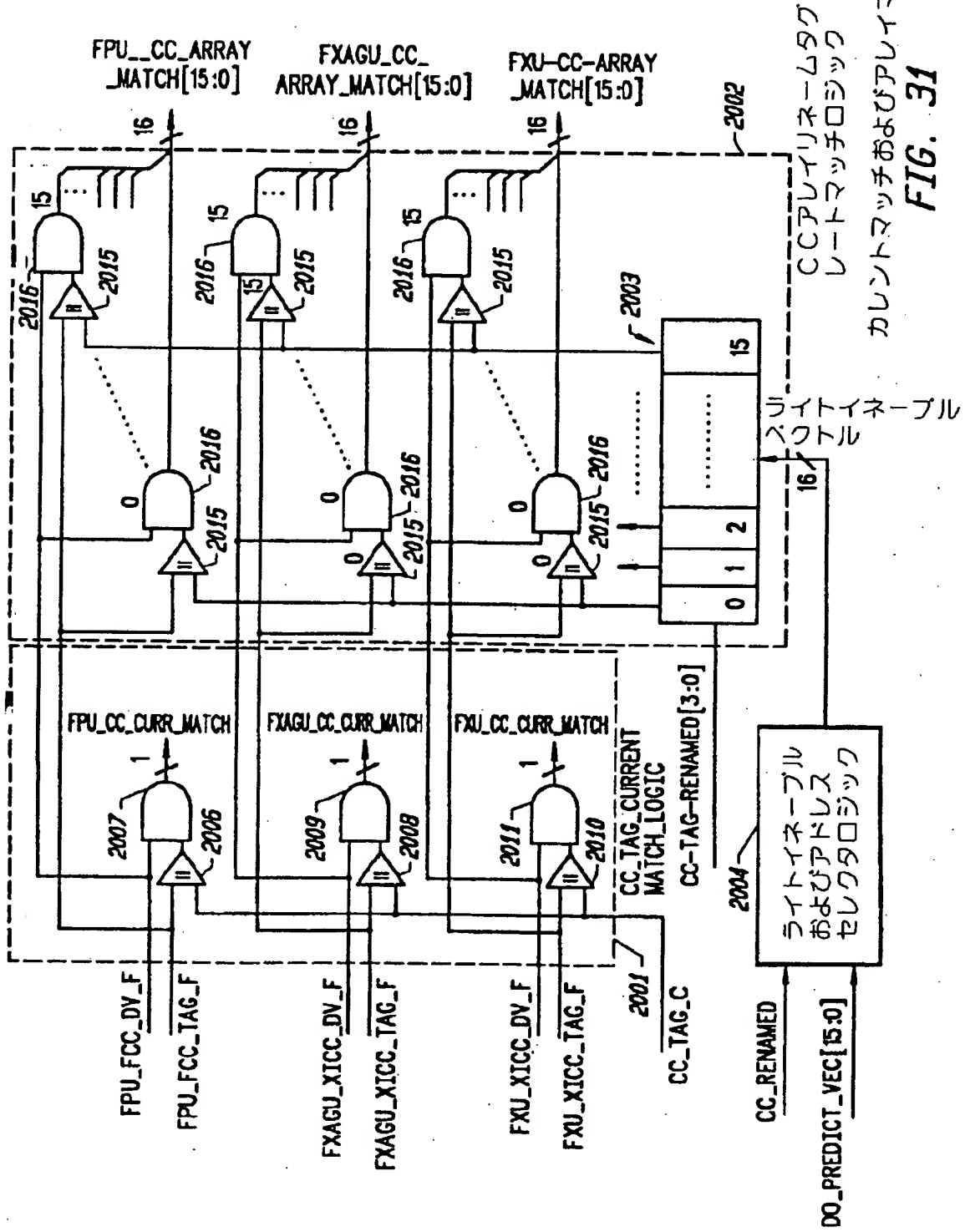


FIG. 31

カレントマッチおよびアレイマッチロジック

CCアレイネームタグ
レートマッチロジックライトイネーブル
およびアドレス
セレクタロジック

DO_PREDICT_VEC[15:0]

CC_RENAMED

CC_TAG_C

CC_TAG_CURRENT
MATCH_LOGICFXU_XICC_DV_F
FXU_XICC_TAG_FFXAGU_XICC_DV_F
FXAGU_XICC_TAG_FFPU_FCC_DV_F
FPU_FCC_TAG_FFPU_CC_ARRAY
MATCH[15:0]

FXAGU_CC_ARRAY_MATCH[15:0]

FXU_CC_ARRAY_MATCH[15:0]

2002

ライトイネーブル
セレクタ

2003

2004

2005

2006

2007

2008

2009

2010

2011

2012

2013

2014

2015

2016

2017

2018

2019

2020

2021

2022

2023

2024

2025

2026

2027

2028

2029

2030

2031

2032

2033

2034

2035

2036

2037

2038

2039

2040

2041

2042

2043

2044

2045

2046

2047

2048

2049

2050

2051

2052

2053

2054

2055

2056

2057

2058

2059

2060

2061

2062

2063

2064

2065

2066

2067

2068

2069

2070

2071

2072

2073

2074

2075

2076

2077

2078

2079

2080

2081

2082

2083

2084

2085

2086

2087

2088

2089

2090

2091

2092

2093

2094

2095

2096

2097

2098

2099

2100

2101

2102

2103

2104

2105

2106

2107

2108

2109

2110

2111

2112

2113

2114

2115

2116

2117

2118

2119

2120

2121

2122

2123

2124

2125

2126

2127

2128

2129

2130

2131

2132

2133

2134

2135

2136

2137

2138

2139

2140

2141

2142

2143

2144

2145

2146

2147

2148

2149

2150

2151

2152

2153

2154

2155

2156

2157

2158

2159

2160

2161

2162

2163

2164

2165

2166

2167

2168

2169

2170

2171

2172

2173

2174

2175

2176

2177

2178

2179

2180

2181

2182

2183

2184

2185

2186

2187

2188

2189

2190

2191

2192

2193

2194

2195

2196

2197

2198

2199

2200

2201

2202

2203

2204

2205

2206

2207

2208

2209

2210

2211

2212

2213

2214

2215

2216

2217

2218

2219

2220

2221

2222

2223

2224

2225

2226

2227

2228

2229

2230

2231

2232

2233

2234

2235

2236

2237

2238

2239

2240

2241

2242

2243

2244

2245

2246

2247

2248

2249

2250

2251

2252

2253

2254

2255

2256

2257

2258

2259

2260

2261

2262

2263

2264

2265

2266

2267

2268

2269

2270

2271

2272

2273

2274

2275

2276

2277

2278

2279

2280

2281

2282

2283

2284

2285

2286

2287

2288

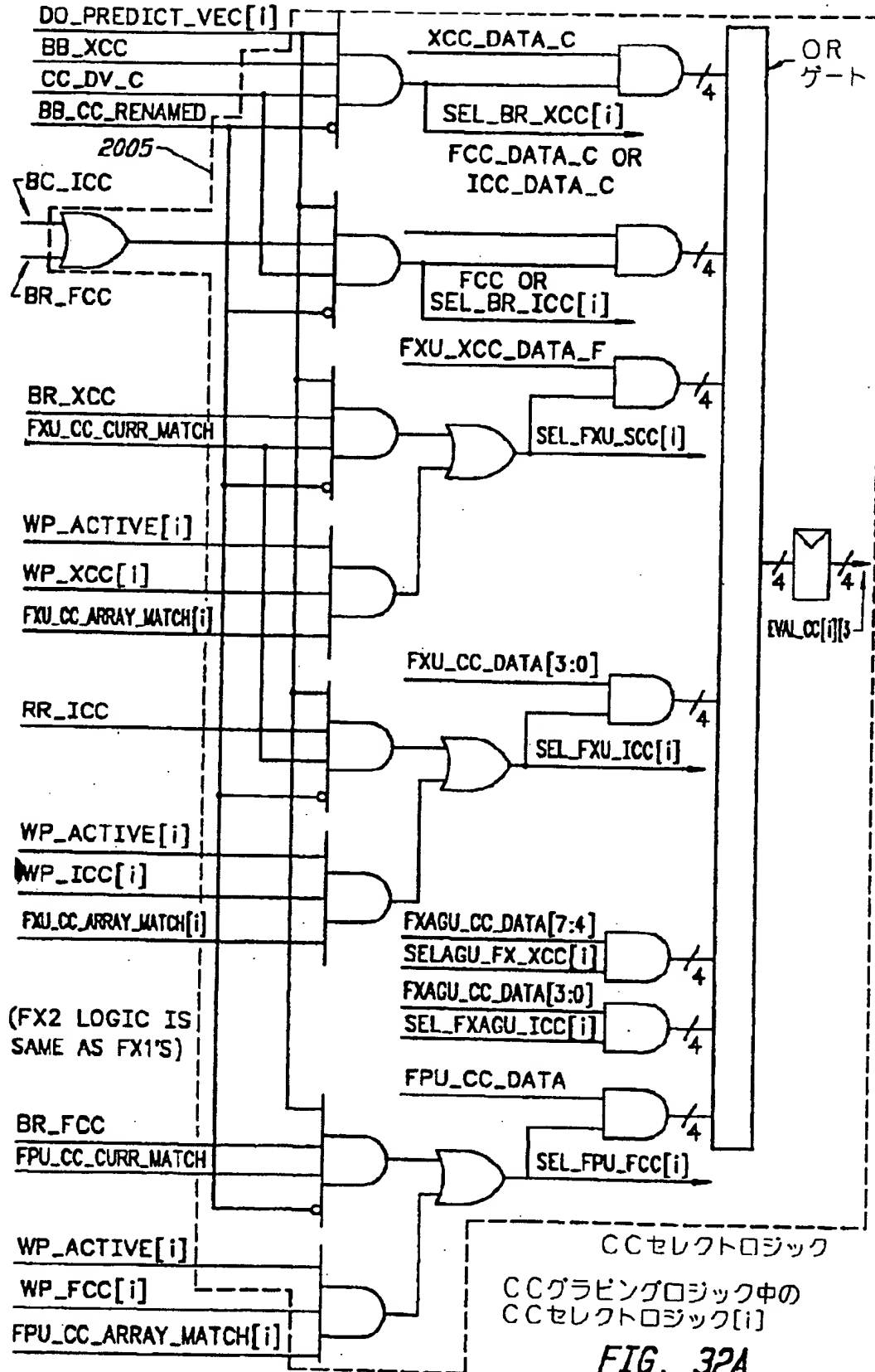
2289

2290

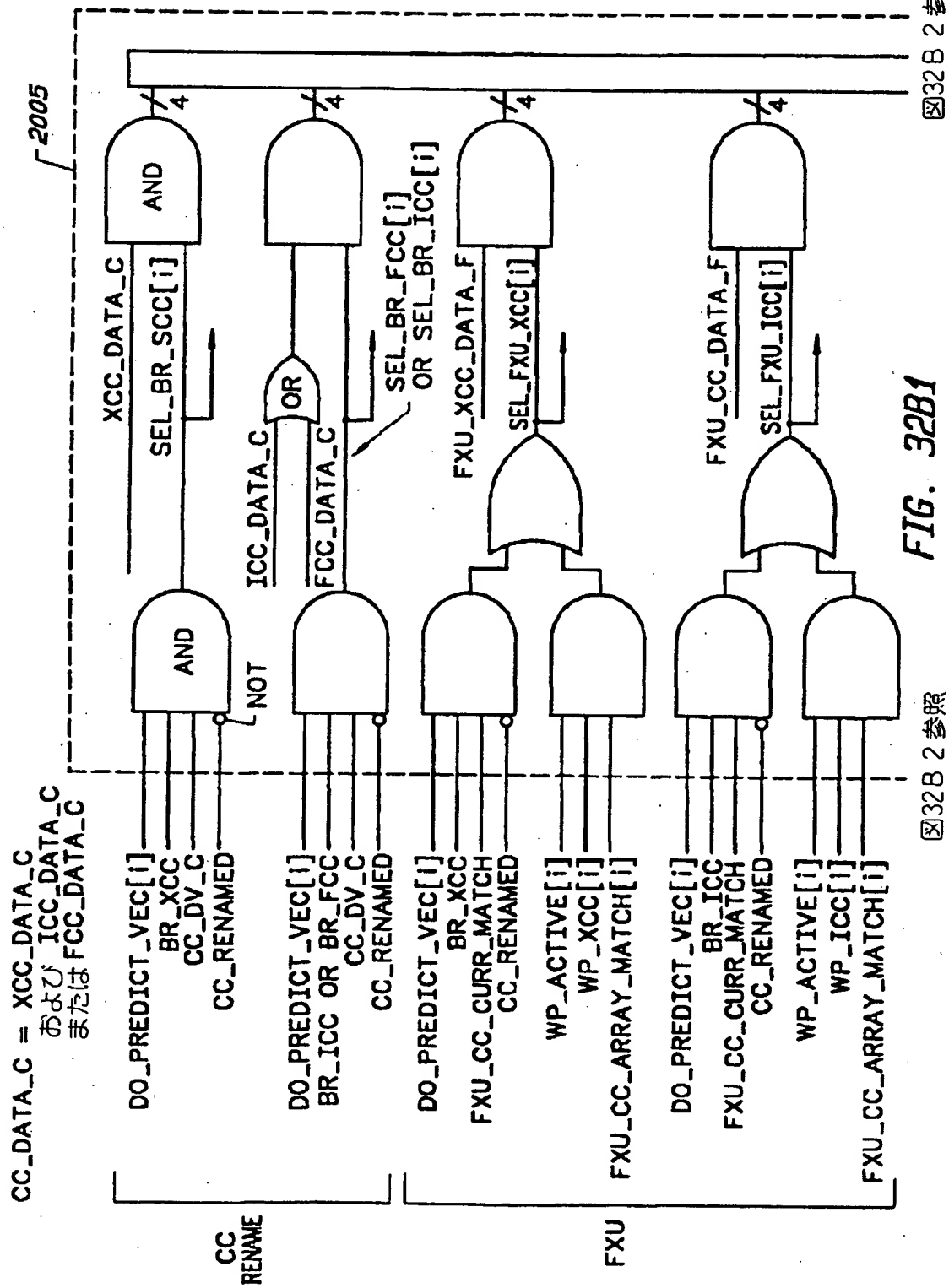
2291

2292

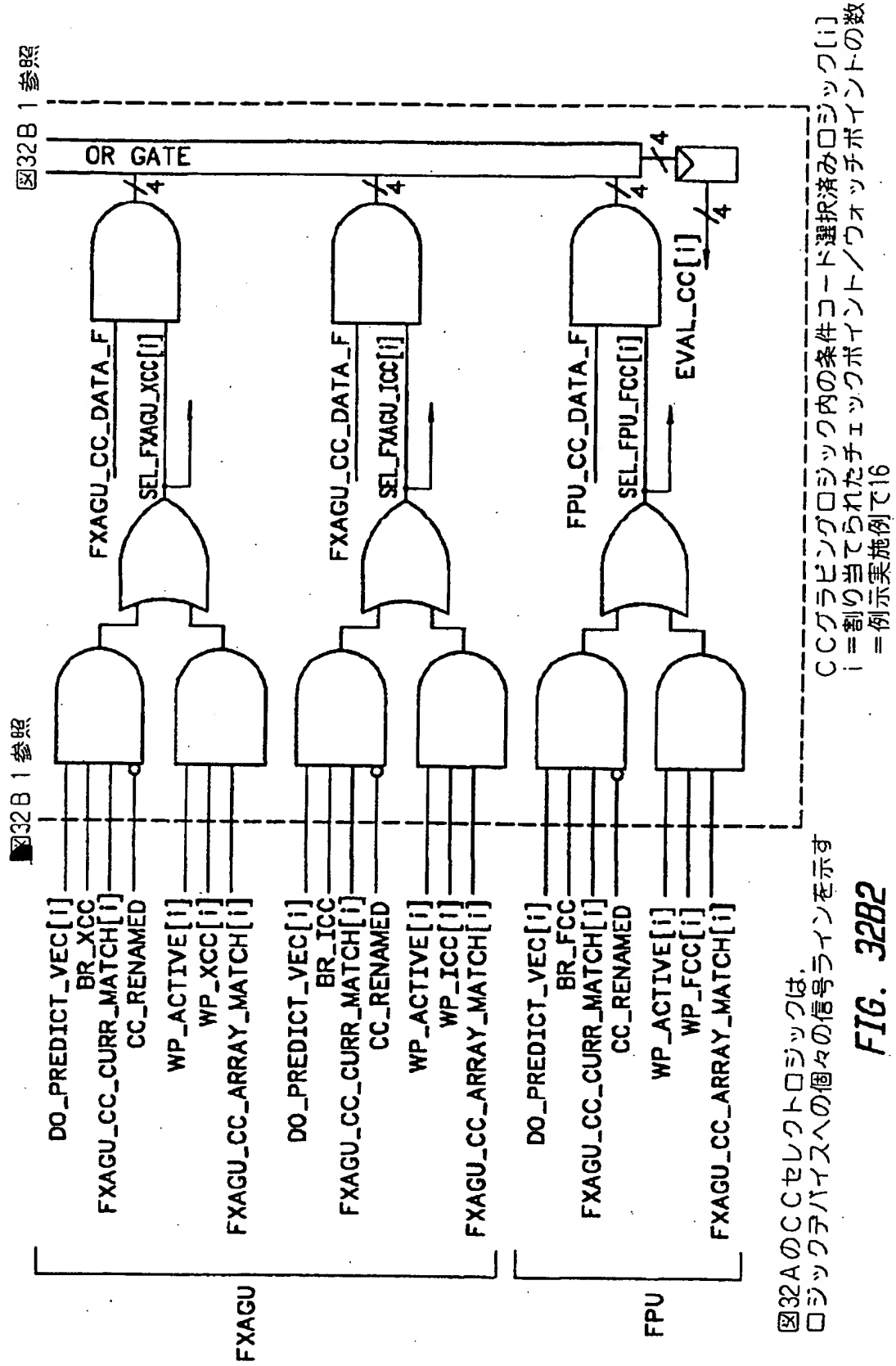
【図32】



【図 3 2】



【 図 3 2 】



【図 3 3】

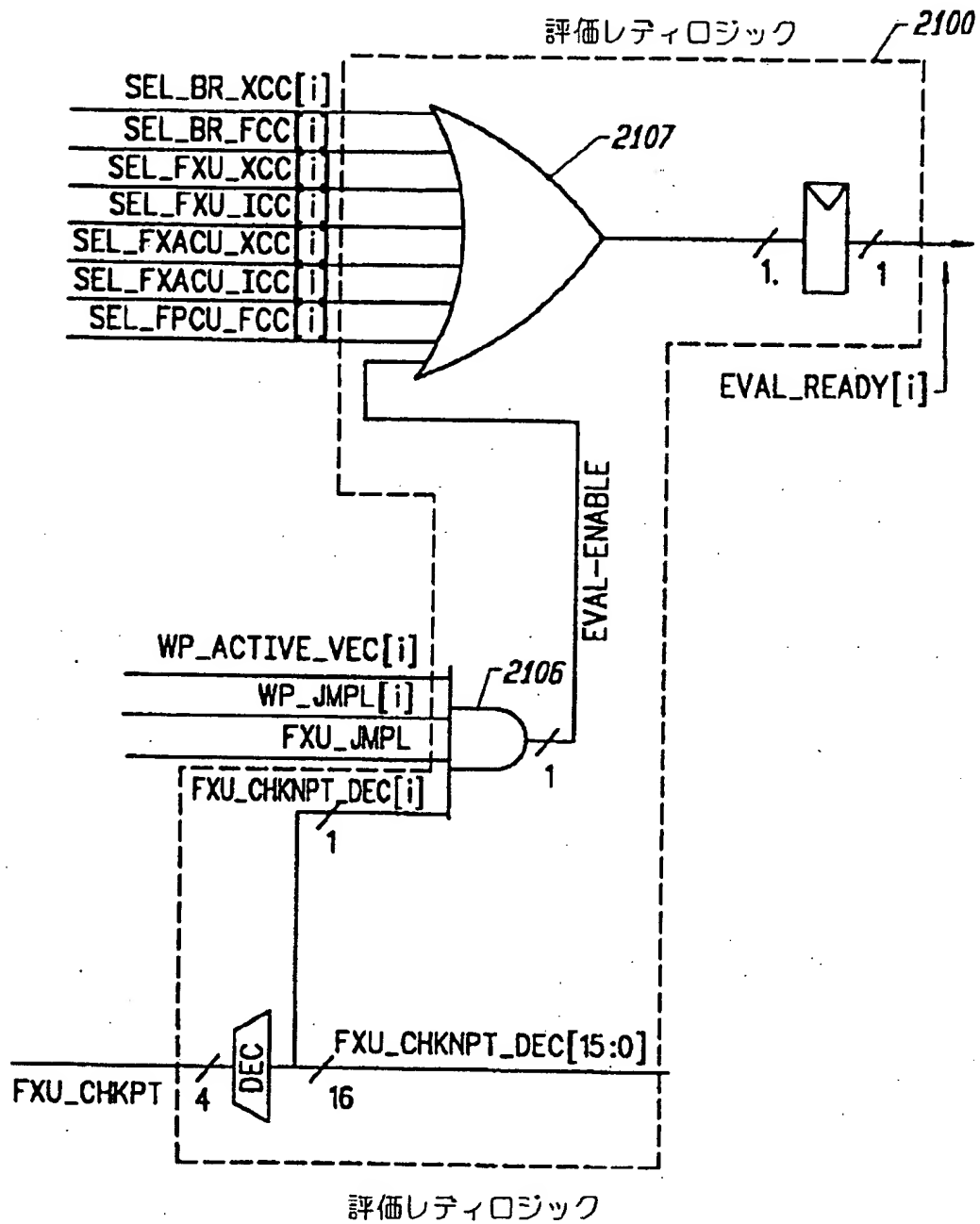
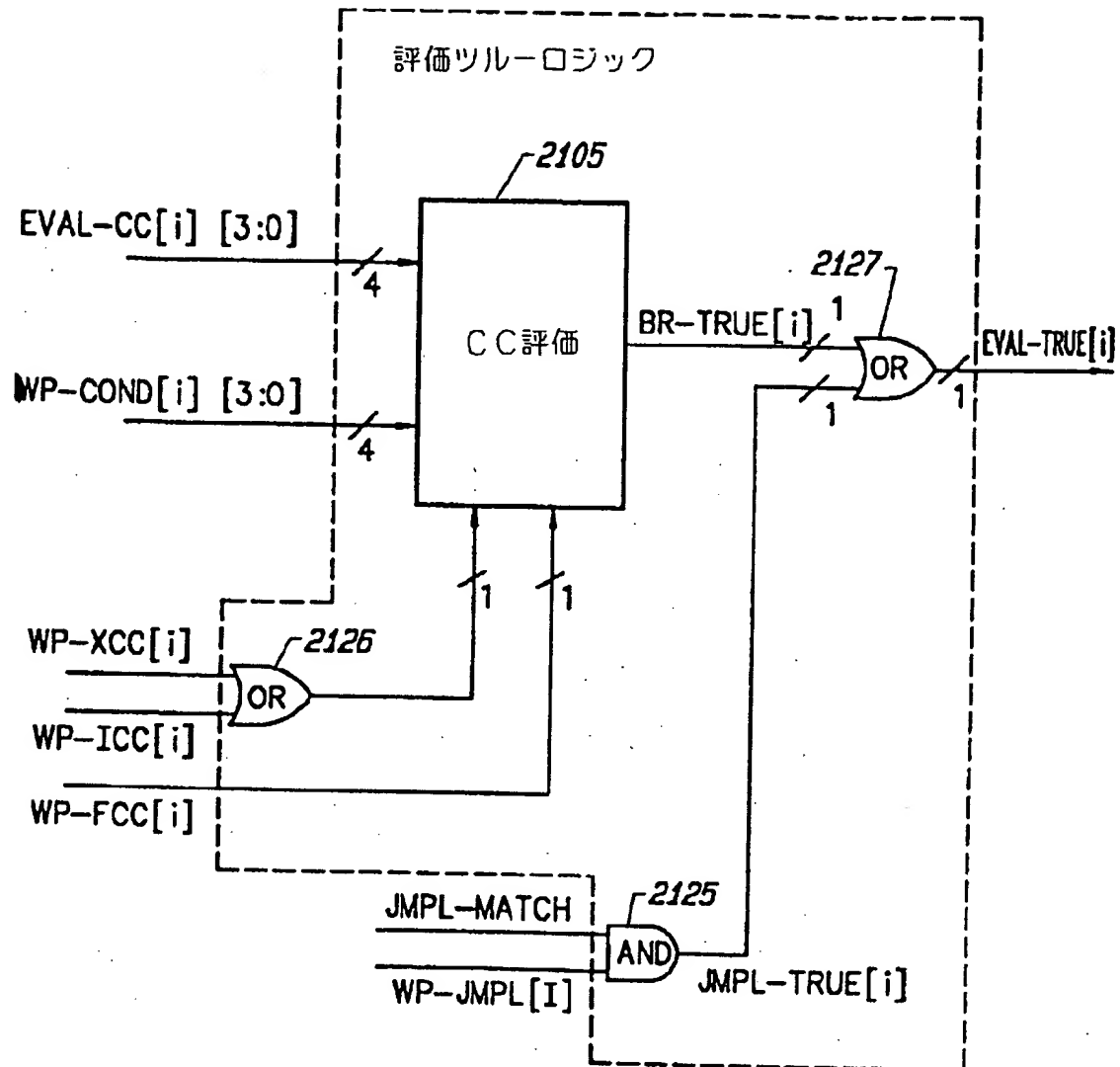


FIG. 33

【図34】



評価ツルーロジック

FIG. 34

【図 3 5】

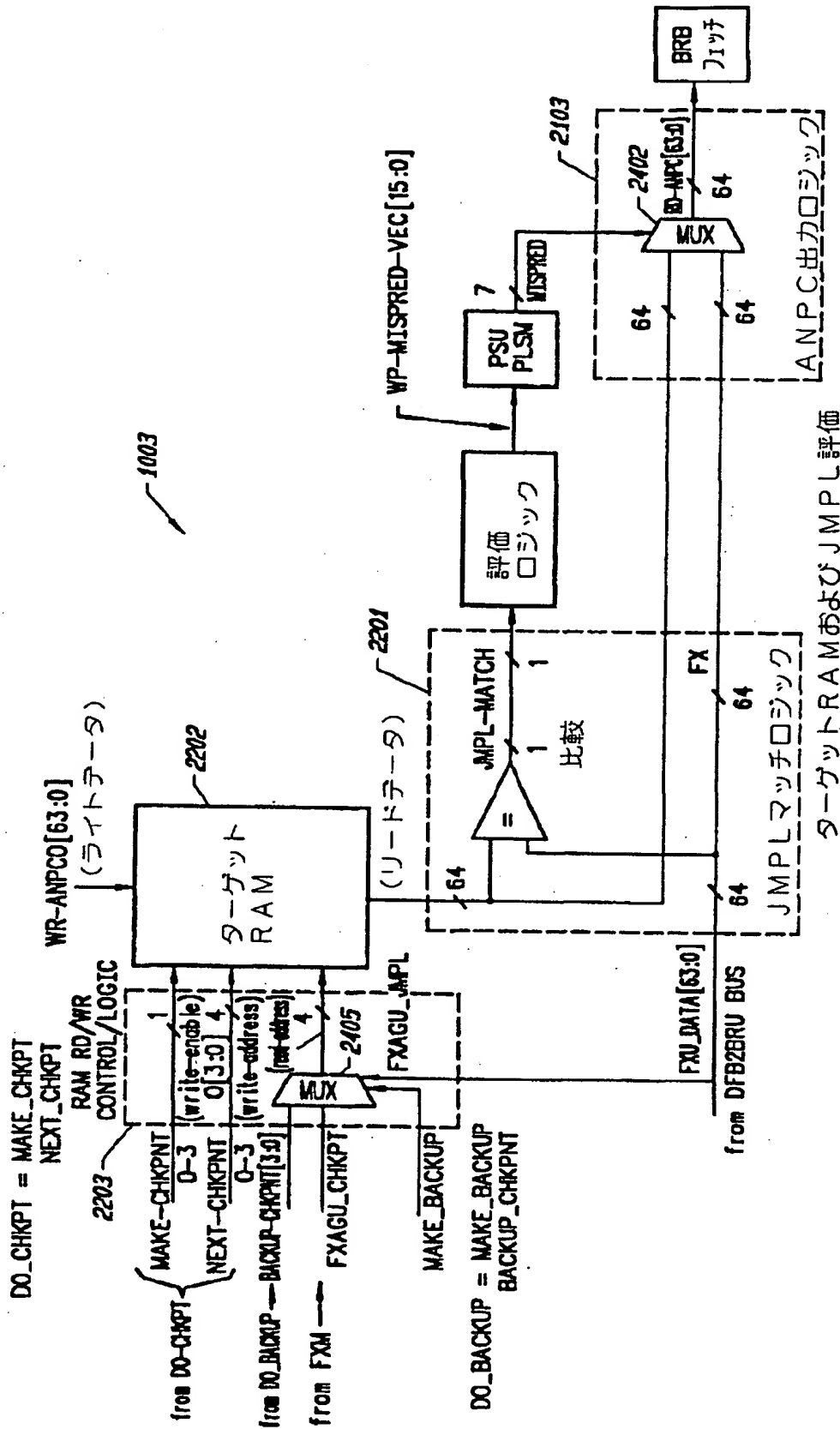
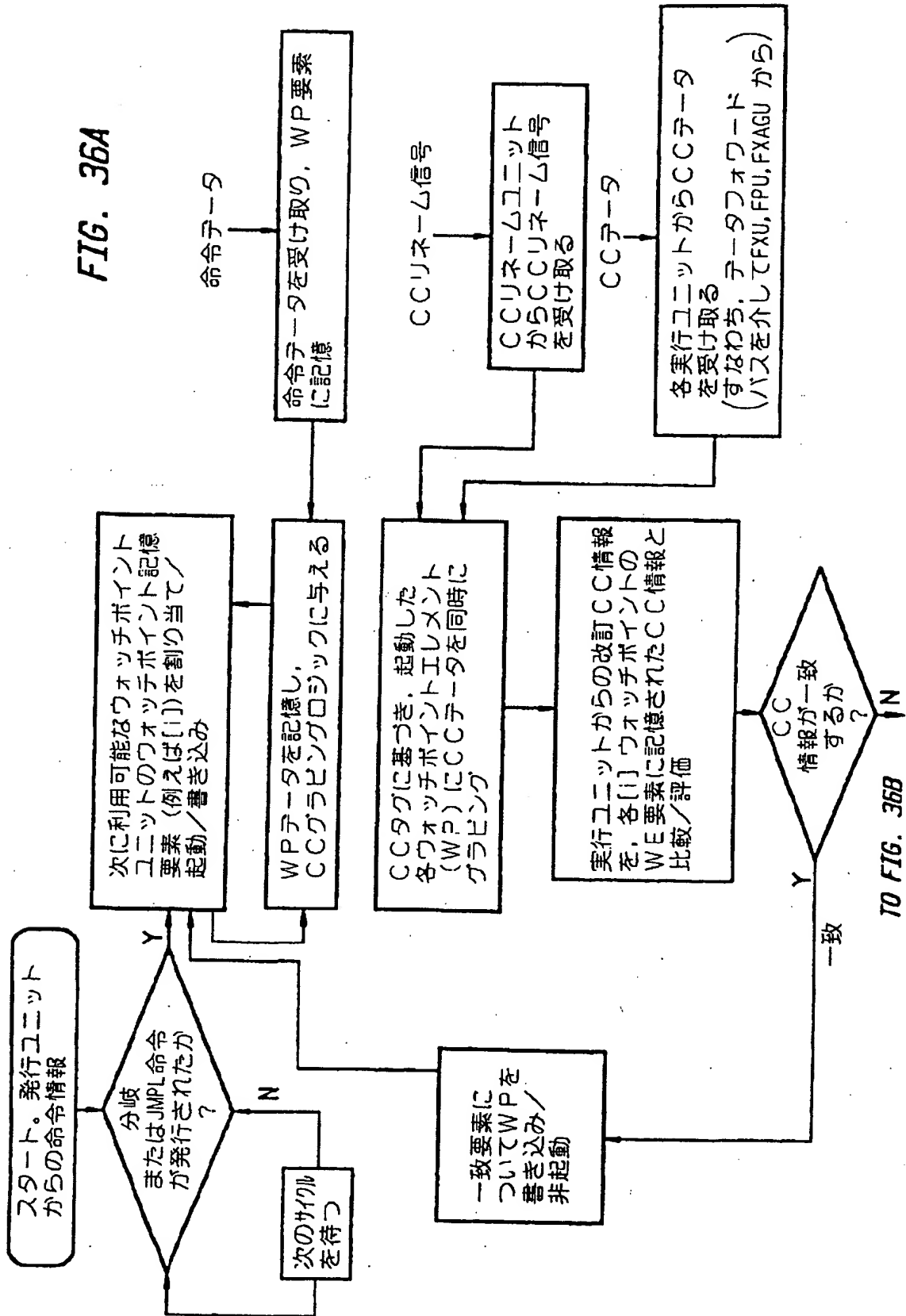


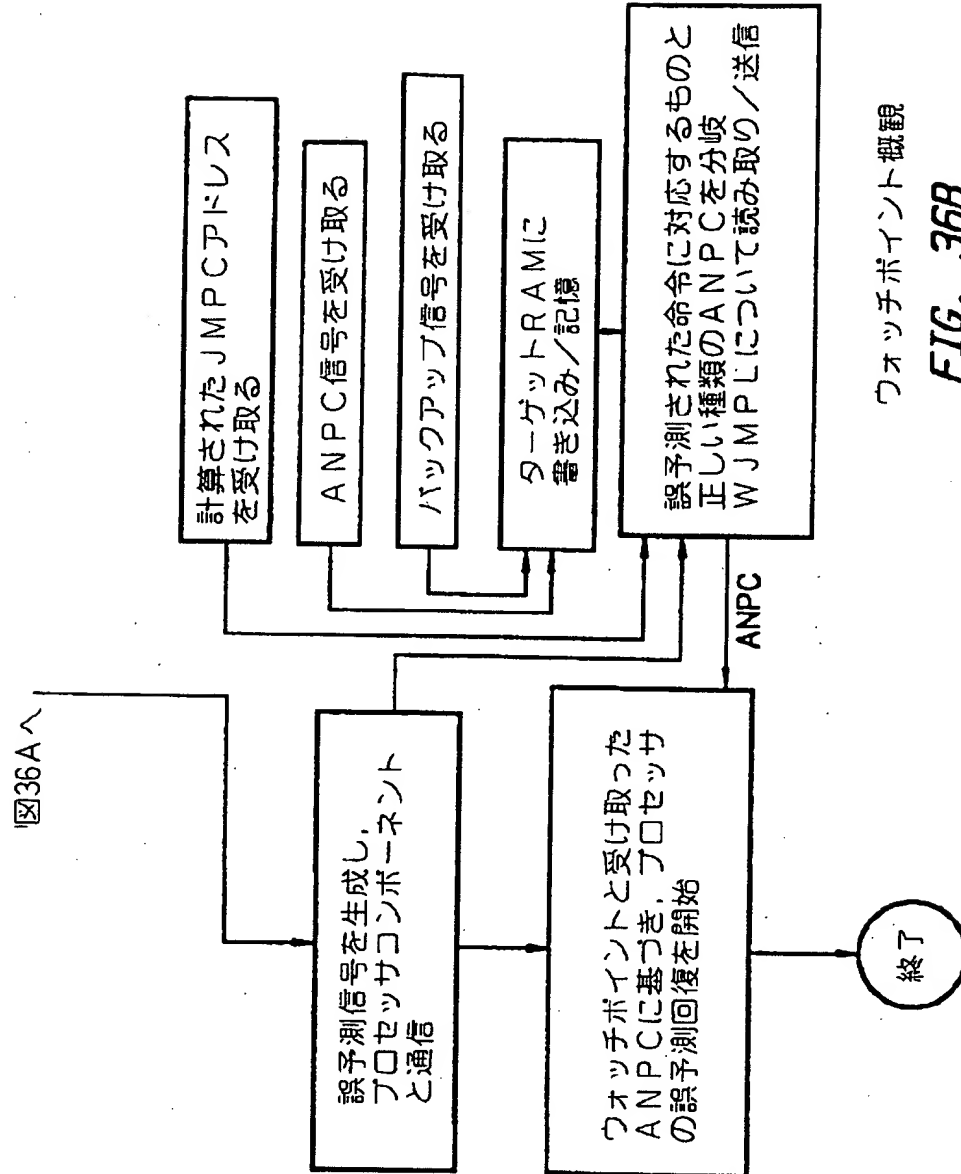
FIG. 35

【図36】

FIG. 36A



【図36】



ウォッチポイント概観

FIG. 36B

【図38】

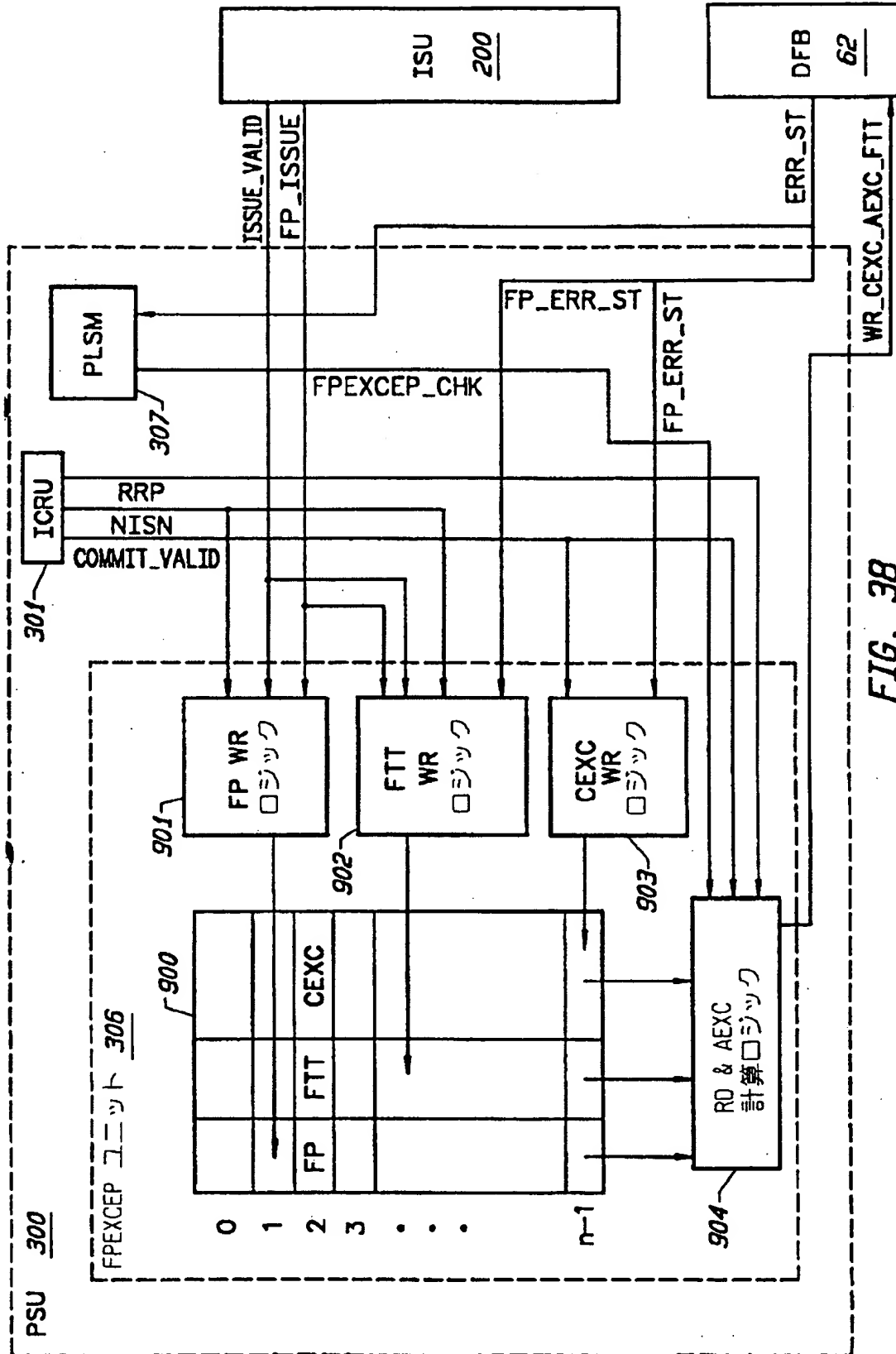


FIG. 38

【図 39】

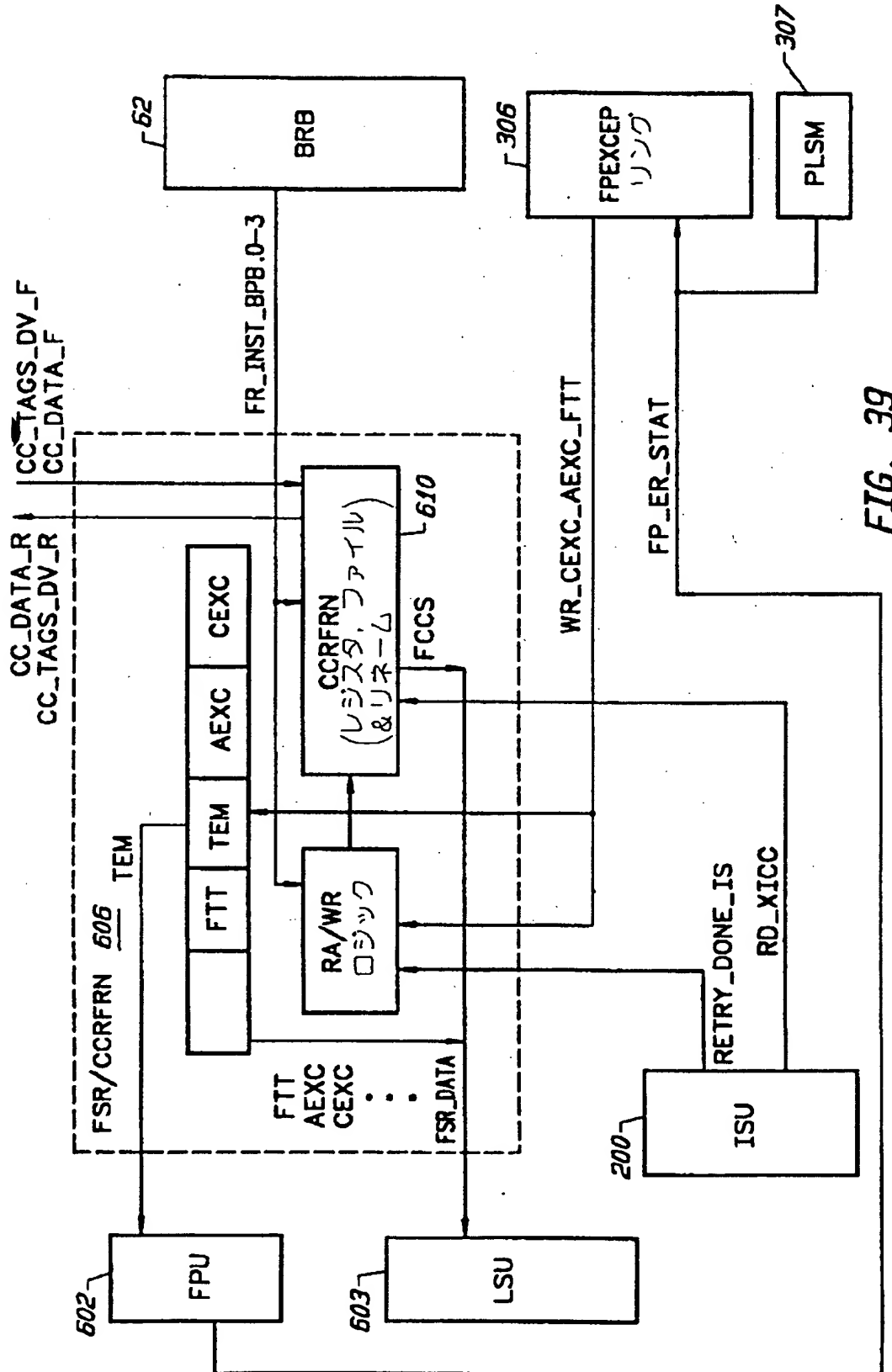


FIG. 39

【図40】

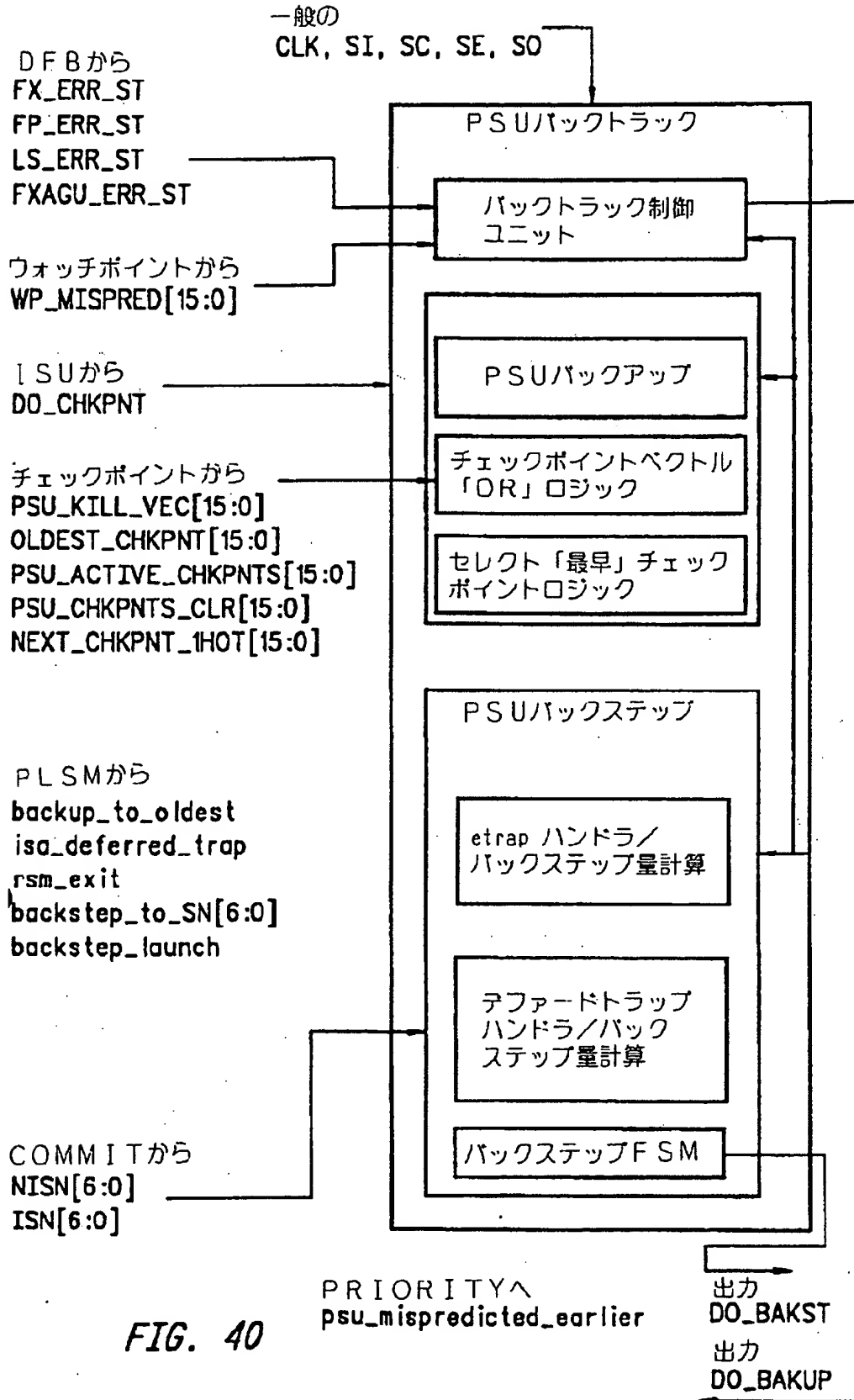


FIG. 40

【 図 4 1 】

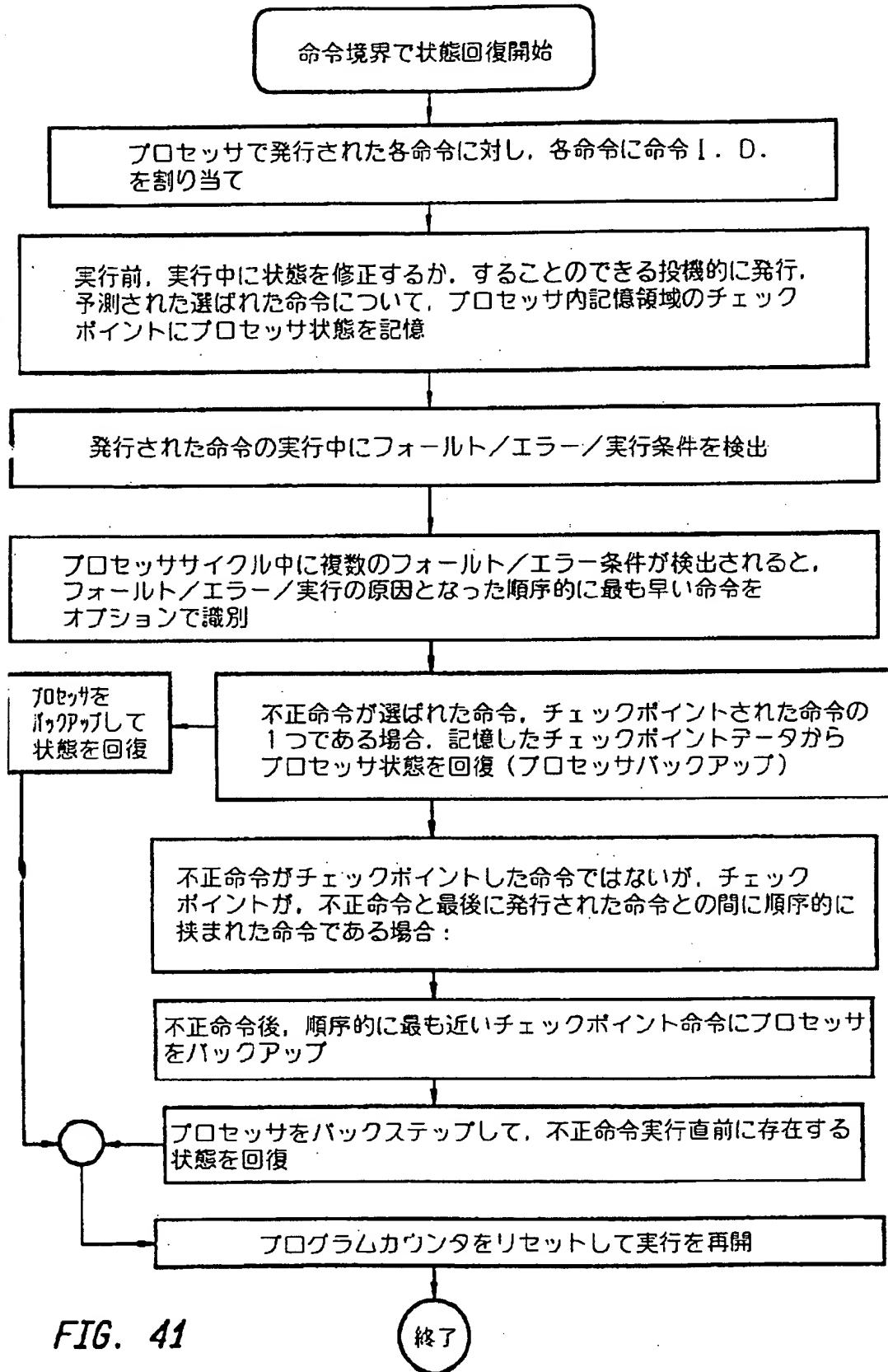


FIG. 41

【図 4 2】

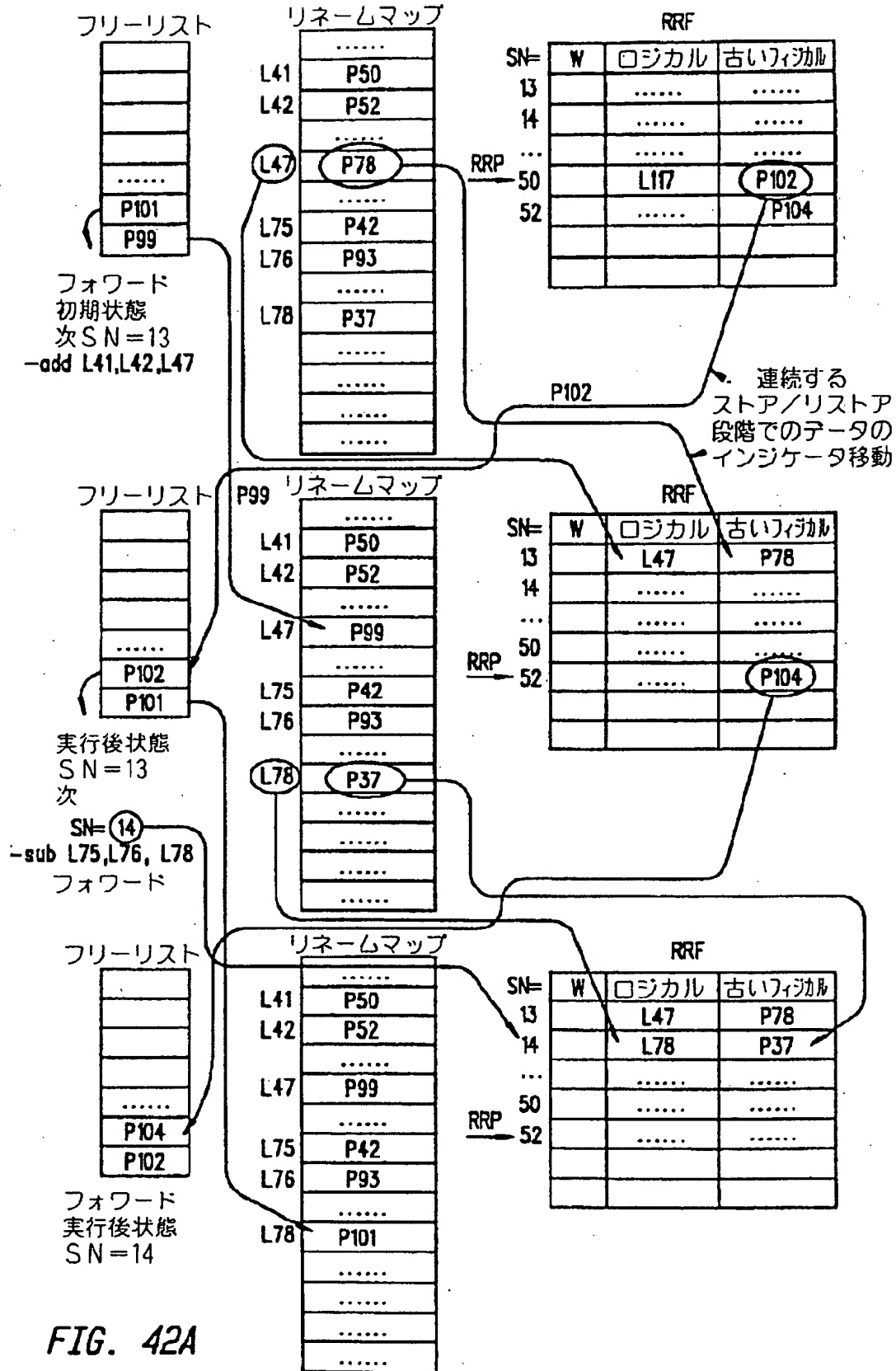
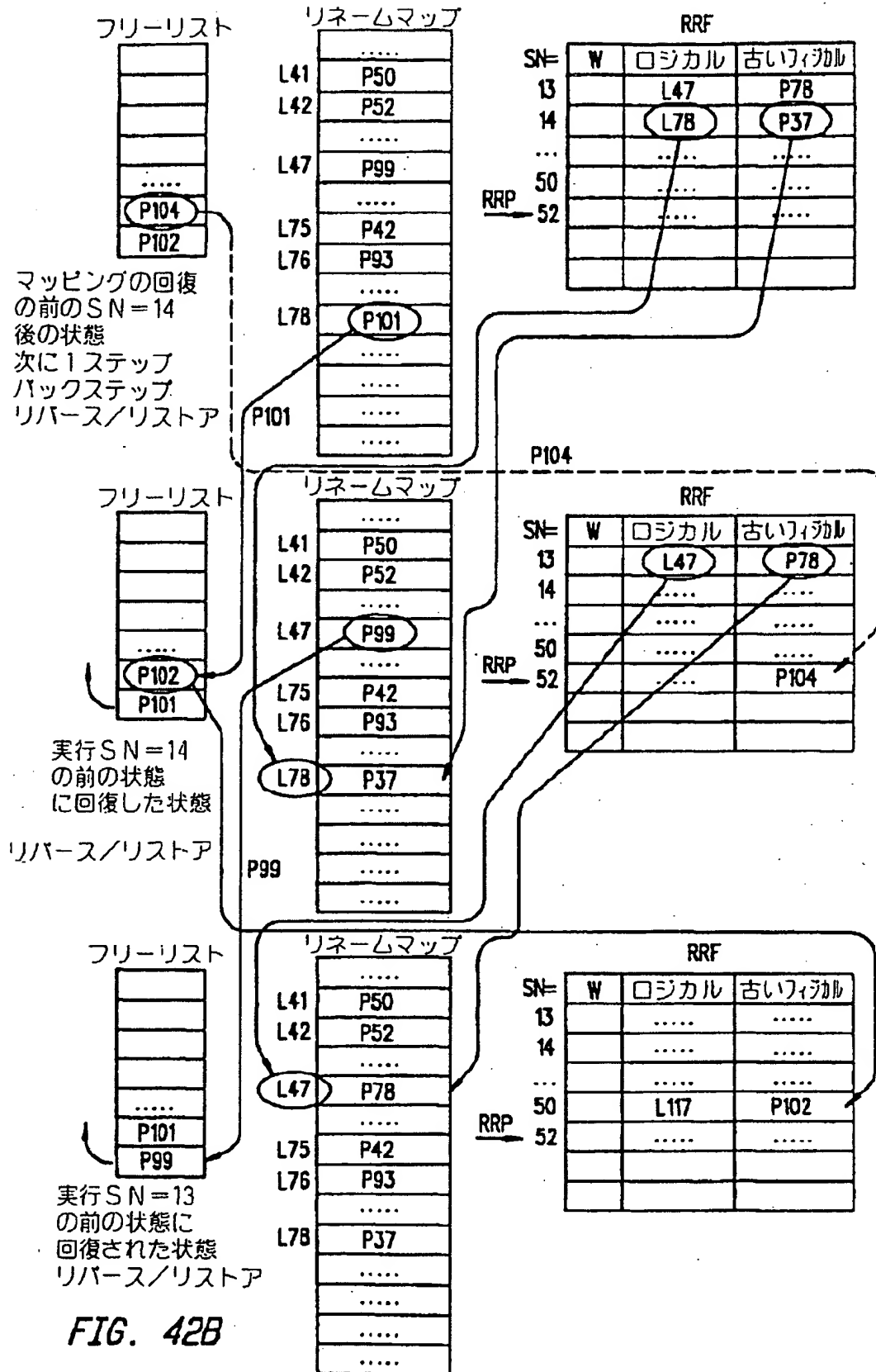
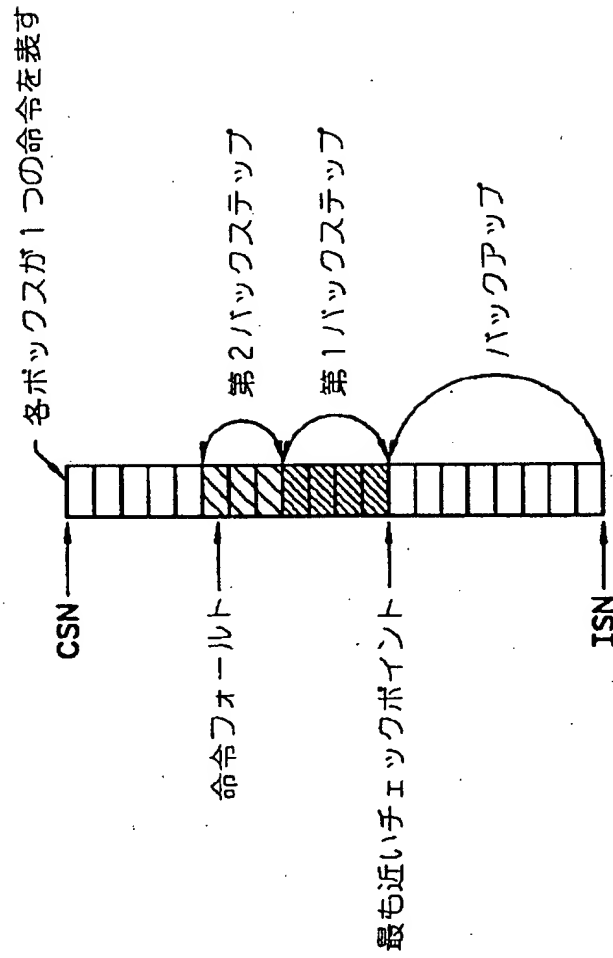


FIG. 42A

【図 4 2】



【図 4 3】



1サイクルあたり4命令の最大バックステップ量を想定し、バックアップの後にバックステップを行う例

FIG. 43

【図 4 4】

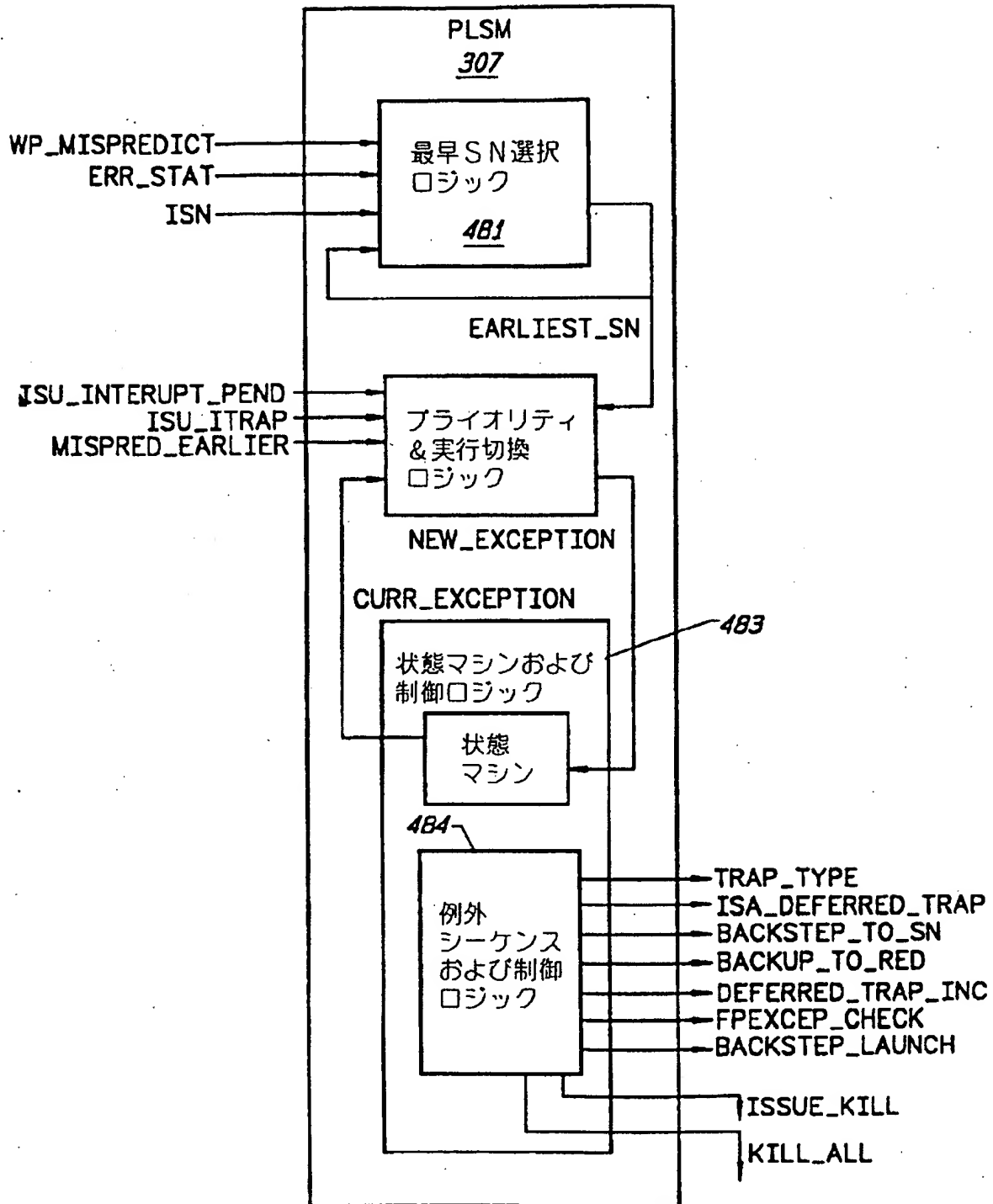
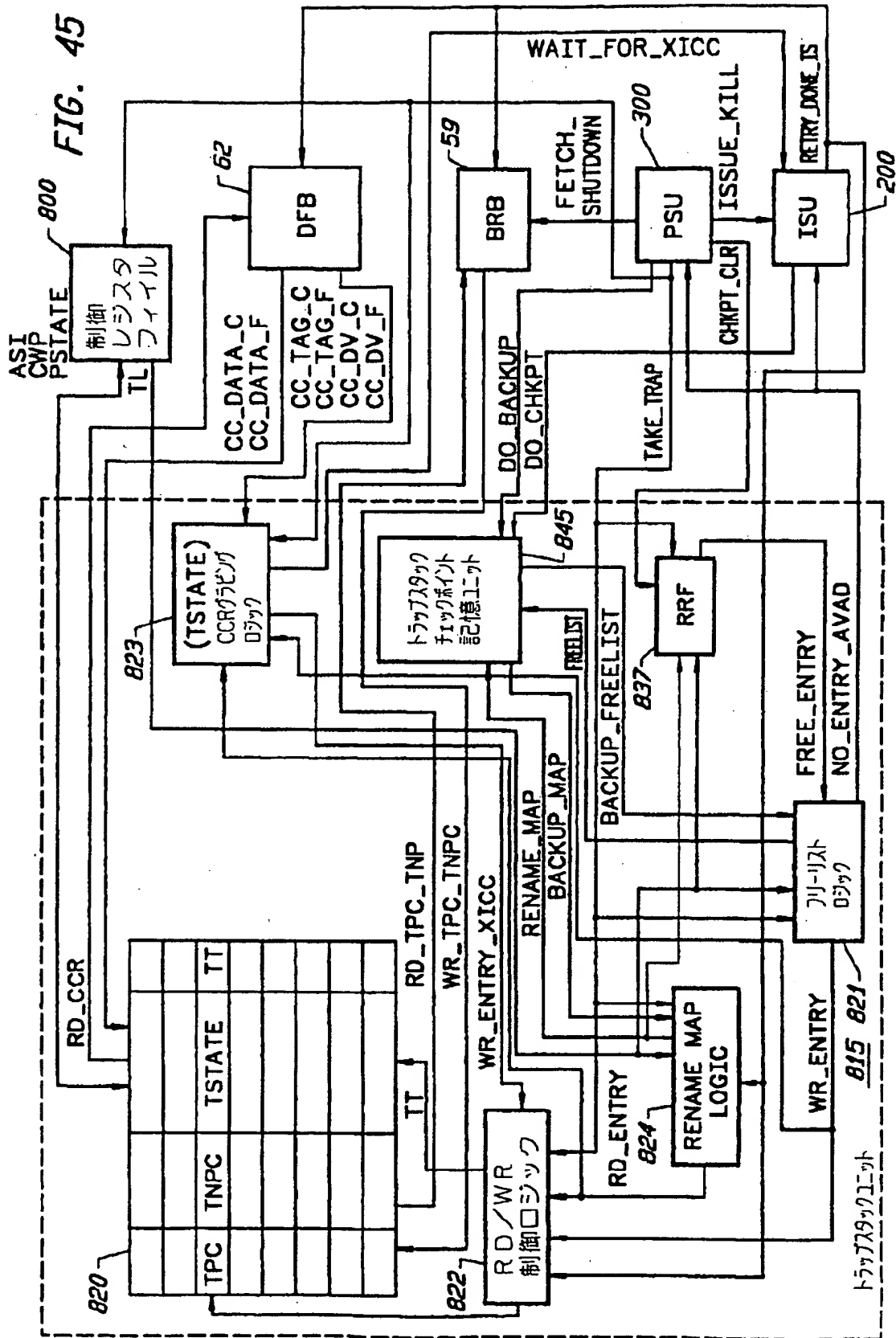


FIG. 44

【図 45】



【図46】

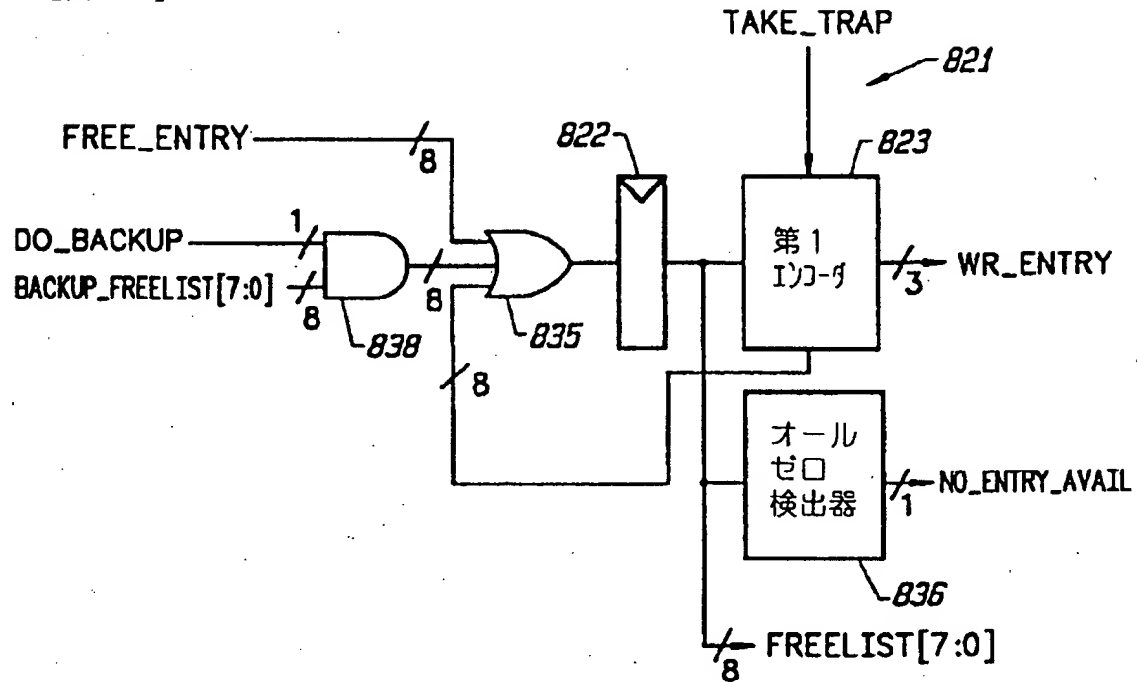


FIG. 46

【図47】

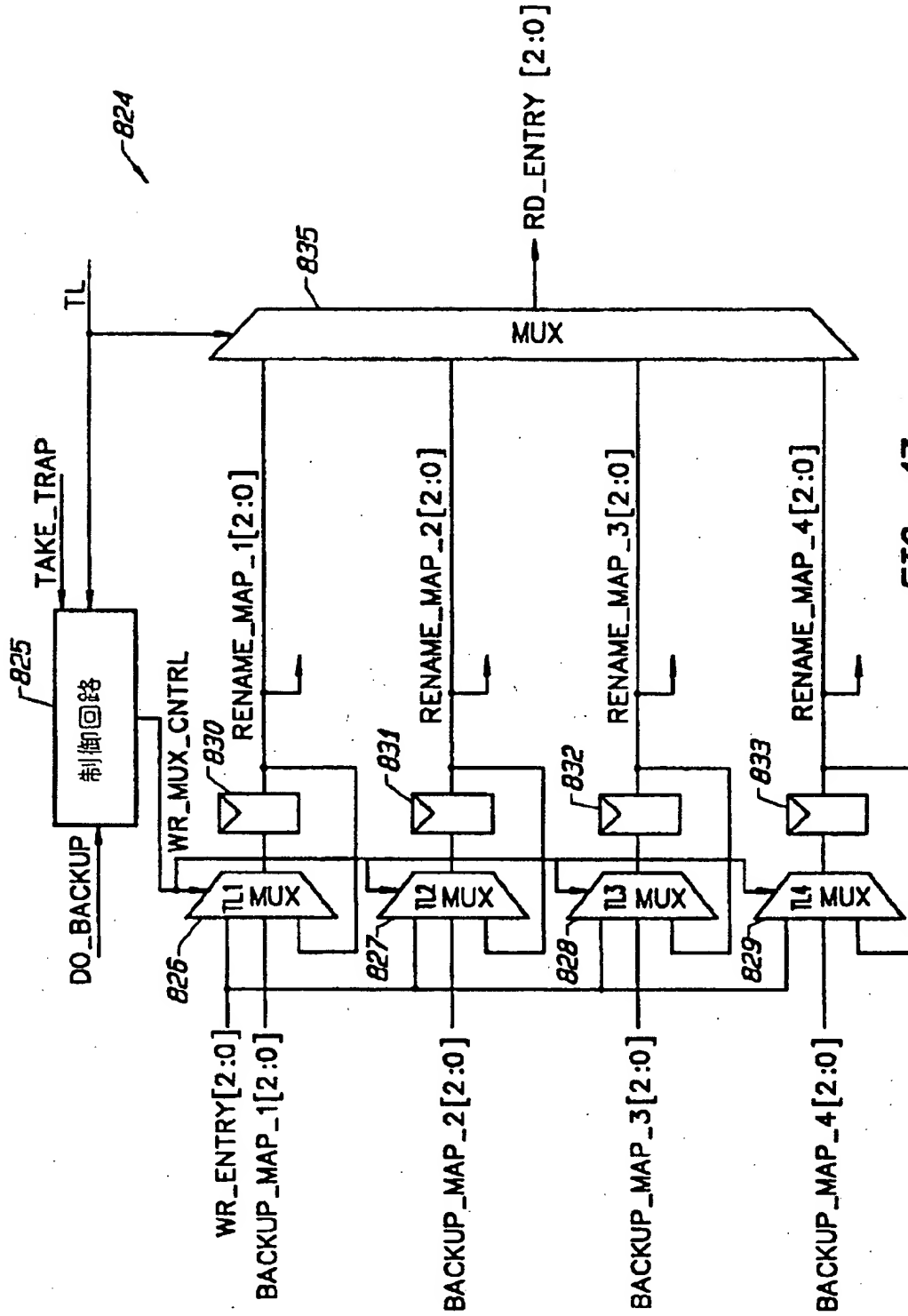


FIG. 47

【図 4 8】

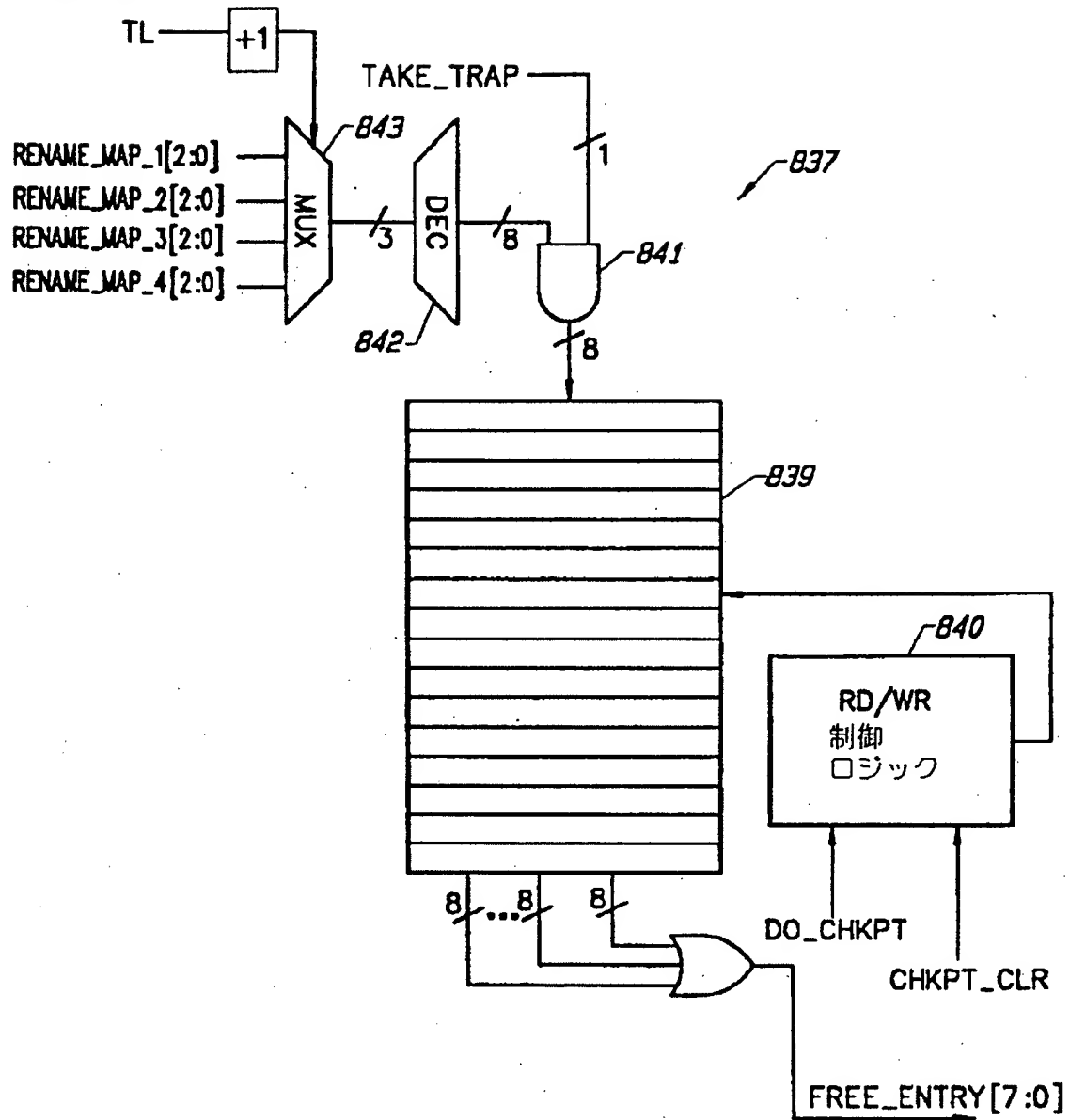


FIG. 48

【図49】

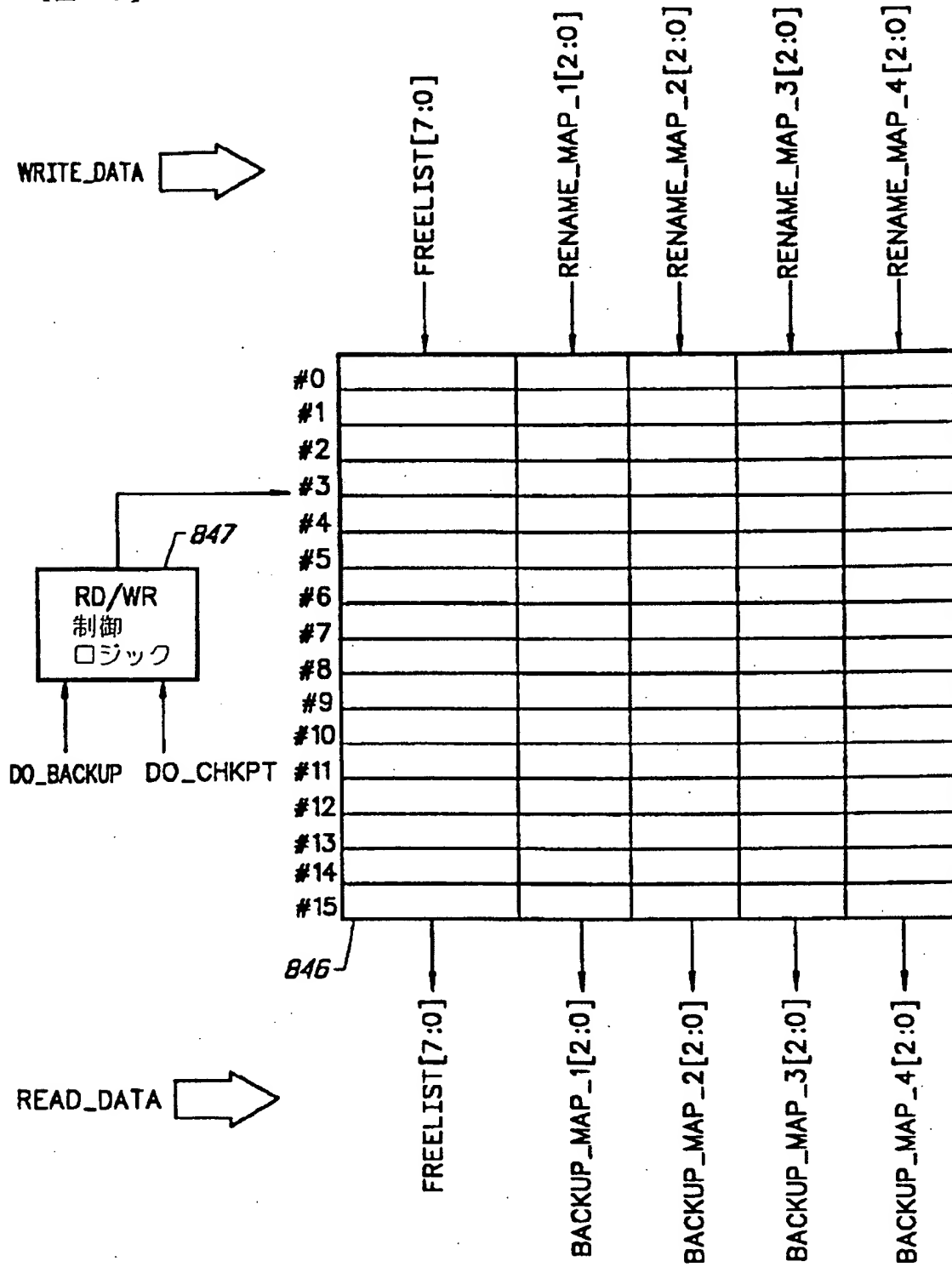


FIG. 49

【図50】

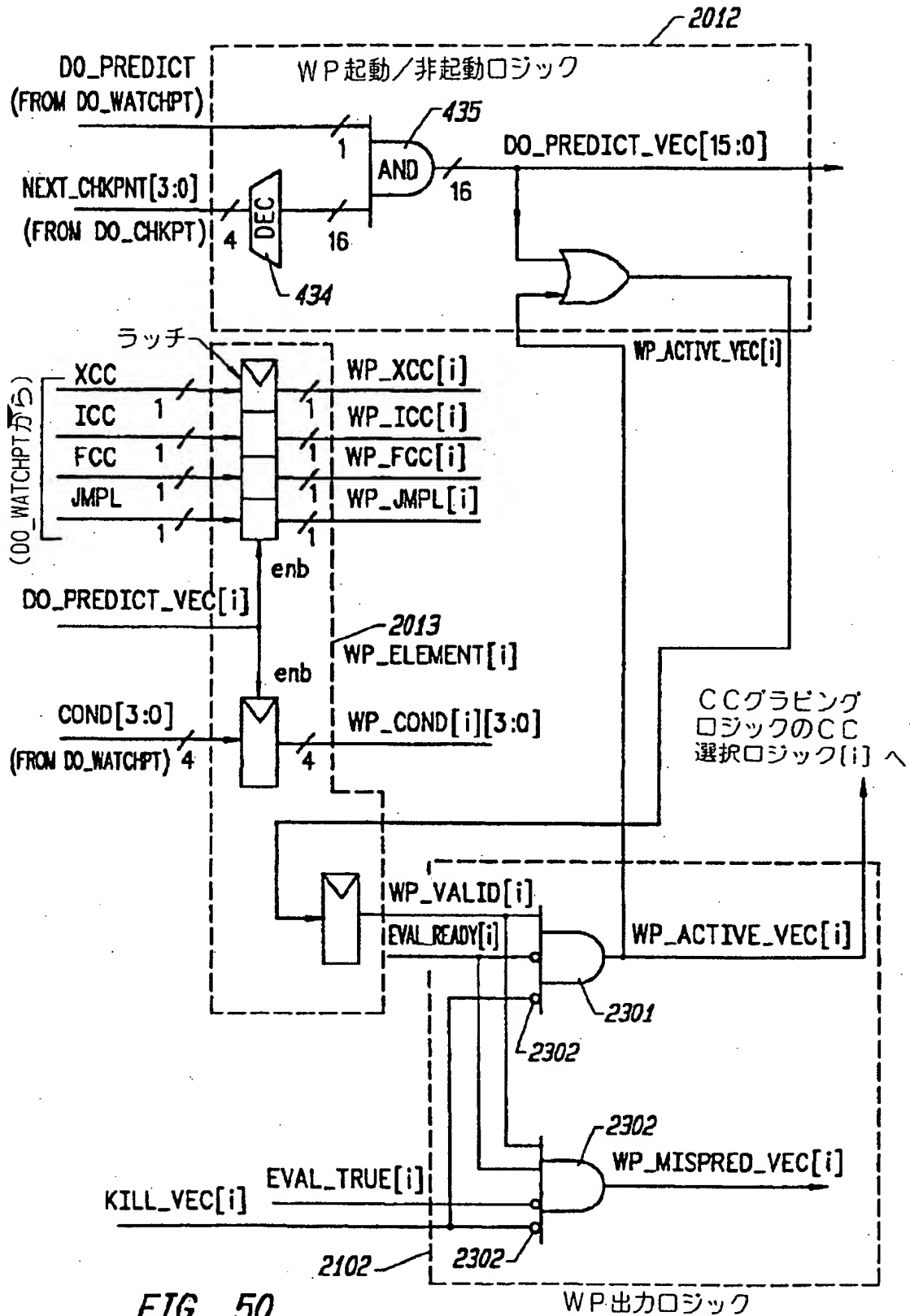


FIG. 50

WPアクティブベクトルおよびWP誤予測ベクトルロジック

【図 5 1】

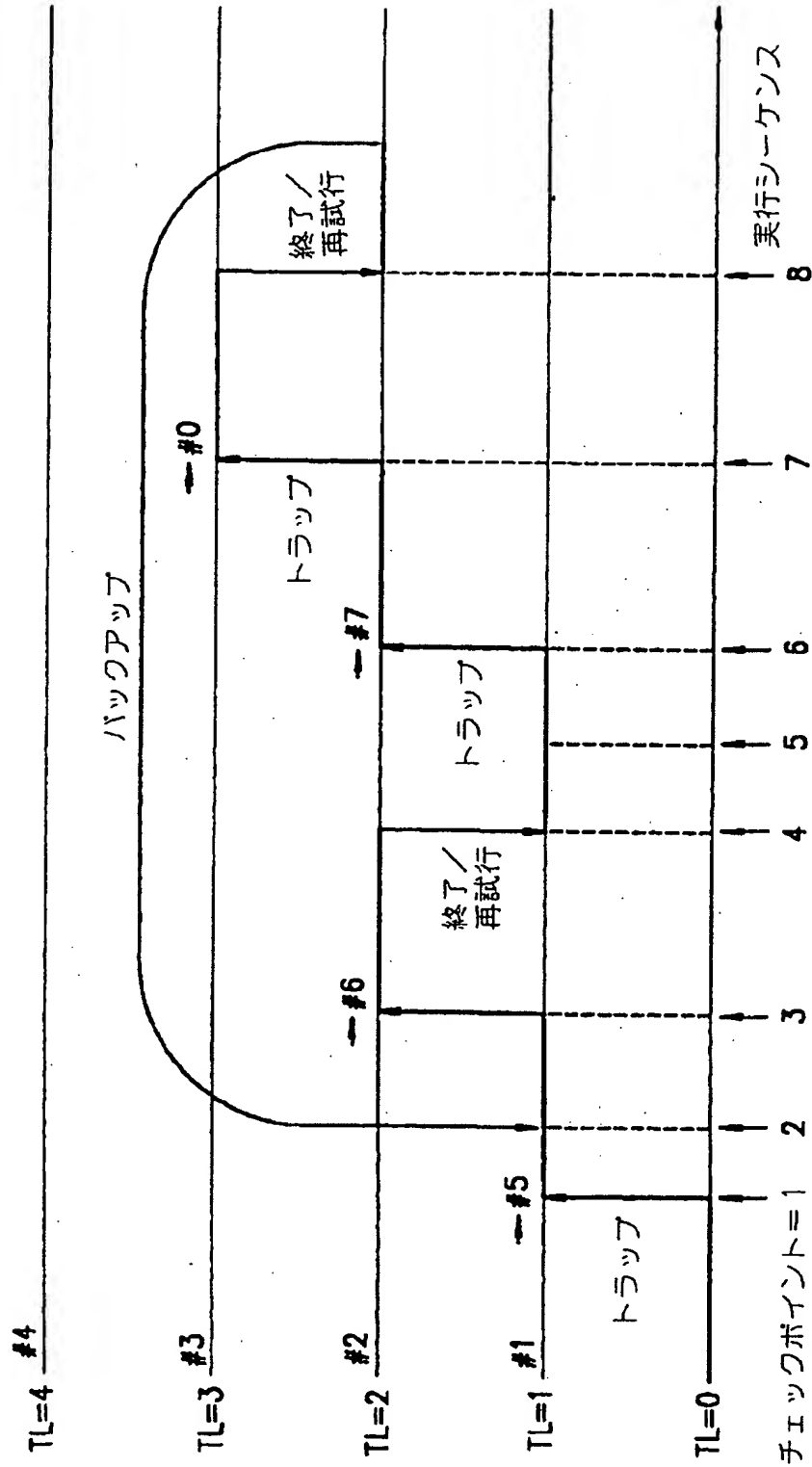
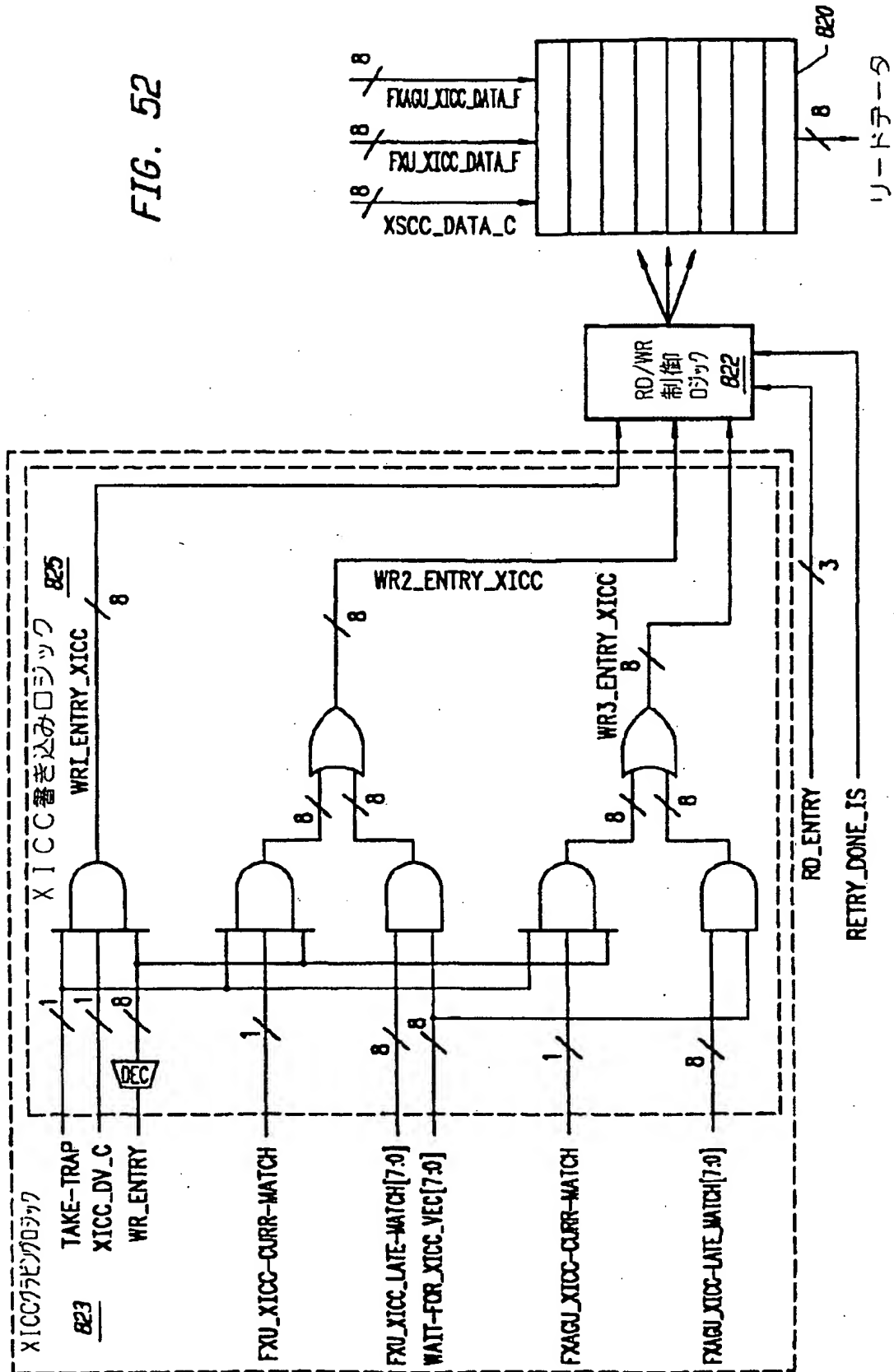


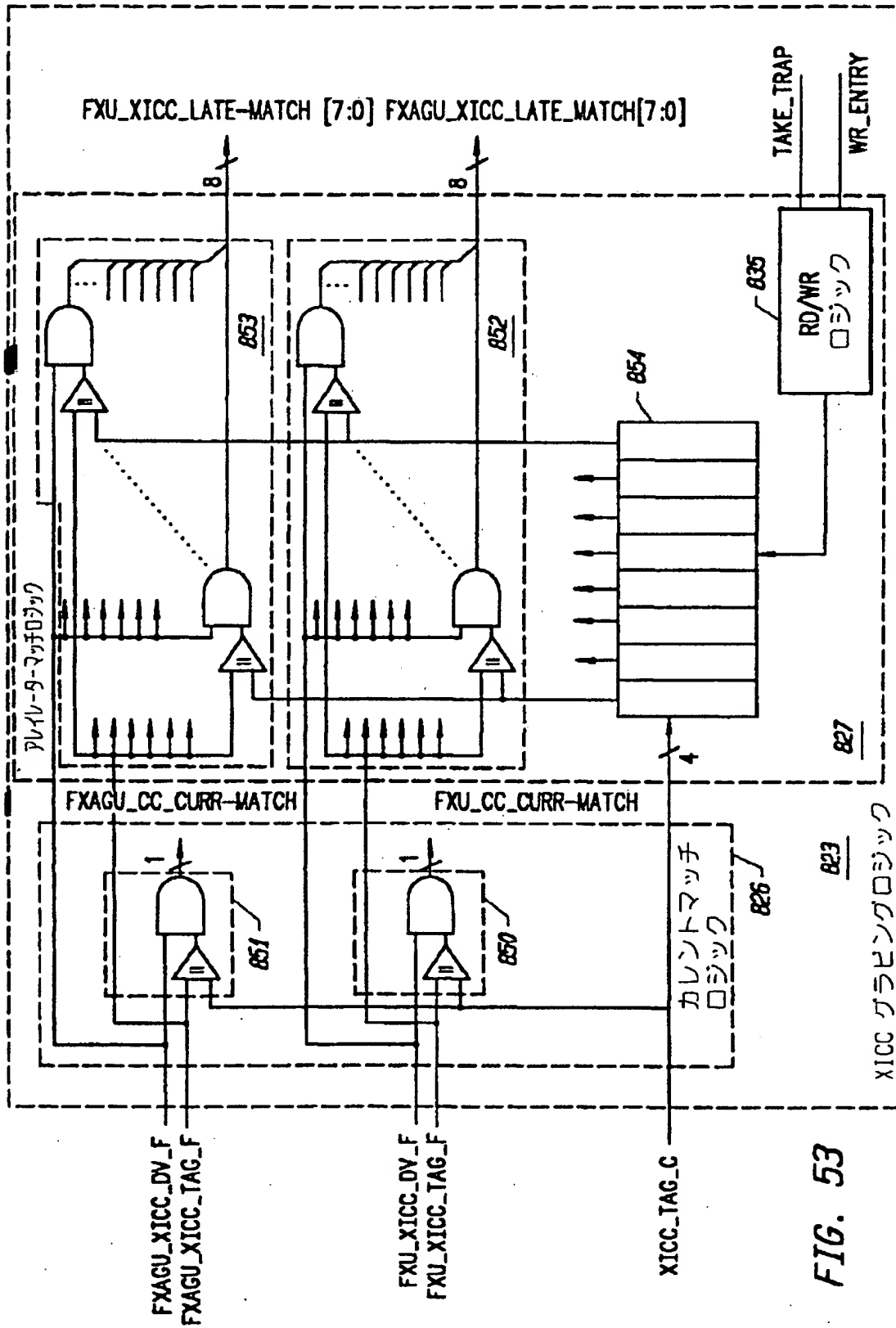
FIG. 51

【図 5 2】

FIG. 52



【図53】



【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/01930

A. CLASSIFICATION OF SUBJECT MATTER IPC(6) : G06F 9/00, 11/00 US CL : 395/182.13-182.19, 733, 375, 800, 650, 700 According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) U.S. : 395/182.13-182.19, 733, 375, 800, 650, 700 Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) PROQUEST		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,E	US, A, 5,497,499 (GARG ET AL.) 05 March 1996, col 1, line 1 through col. 7, line 15.	1-33, 37-39, 62-76, 78-92, 109-110, 139-160, 163-165, 192-194, 199, and 219-236
Y,P	US, A, 5,471,598 (QUATTROMANI ET AL.) 28 November 1995, col. 4, line 40 through col. 27, line 17.	40-61, 77, 108, 128-138, 166-187, 195-198, 200, and 237-245
<input checked="" type="checkbox"/> Further documents are listed in the continuation of Box C. <input type="checkbox"/> See patent family annex.		
* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be part of particular relevance "E" earlier document published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reasons (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "Z" document member of the same patent family		
Date of the actual completion of the international search 22 MAY 1996		Date of mailing of the international search report 14 JUN 1996
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230		Authorized officer William M. Treat Telephone No. (703) 308-7613

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/01930

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US, A, 5,269,017 (HAYDEN ET AL.) 07 December 1993, col. 1, line 1 through col. 12, line 56.	94-107, 111, 161-162, and 188-191
Y	US, A, 5,355,457 (SHEBANOW ET AL.) 11 October 1994, col 3, lines 23-37	94-107, 111, 161-162, and 188-191
Y,P	US, A, 5,463,745 (VIDWANS ET AL.) 31 October 1995, col. 5, line 50 through col. 24, line 61.	34-36
Y,P	US, A, 5,481,685 (NGUYEN ET AL.) 02 January 1996, col. 6, line 5 through col. 21, line 5 and col. 25, line 39 through col. 54, line 65	201-218

INTERNATIONAL SEARCH REPORT

Int. national application No.
PCT/US96/01930

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This international report has not been established in respect of certain claims under Article 17(C)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:
2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:
3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

Please See Extra Sheet.

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:
4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest



The additional search fees were accompanied by the applicant's protest.



No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US96/01930

BOX II. OBSERVATIONS WHERE UNITY OF INVENTION WAS LACKING

This ISA found multiple inventions as follows:

Group I, claims 1-33, 37-39, 62-76, 78-92, 109-110, 139-160, 163-165, 192-194, 199, and 219-236, drawn to a system for tracking speculative instruction execution/status.

Group II, claims 40-61, 77, 108, 128-138, 166-187, 195-198, 200, and 237-245, drawn to a system for scheduling long latency instructions ahead of low latency instructions.

Group III, claims 94-107, 111, 161-162, 188-191, drawn to a system for checkpointing an instruction to reduce the amount of checkpointed data.

Group IV, claims 34-36, drawn to a system for issuing instructions based on resource availability.

Group V, claim 93, drawn to a system for designing optimum checkpoint size.

Group VI, claims 201-216, drawn to a system for exception processing, Group VII, claims 112-127, drawn to a system for bounding exception handling.

Group I contains the special technical feature of a system for tracking speculative instruction execution, which is not found in Groups II-VII.

Group II contains the special technical feature of a system for scheduling long latency instructions ahead of low latency instructions, which is not found in Groups I and III-VII.

Group III contains the special technical feature of a system for checkpointing an instruction to reduce the amount of checkpointed data, which is not found in Groups I-II and IV-VII. Group IV contains the special technical feature of a system for issuing instructions based on resource availability, which is not found in Groups I-III and V-VII.

Group V contains the special technical feature of a system for designing optimum checkpoint size, which is not found in Groups I-IV and VI-VII.

Group VI contains the special technical feature of a system for exception processing, which is not found in Groups I-V and VII.

Group VII contains the special technical feature of a system for bounding exception handling, which is not found in Groups I-VI.

フロントページの続き

- (31)優先権主張番号 08/472, 394
(32)優先日 1995年6月7日
(33)優先権主張国 米国 (US)
- (31)優先権主張番号 08/473, 223
(32)優先日 1995年6月7日
(33)優先権主張国 米国 (US)
- (31)優先権主張番号 08/476, 419
(32)優先日 1995年6月7日
(33)優先権主張国 米国 (US)
- (31)優先権主張番号 08/478, 025
(32)優先日 1995年6月7日
(33)優先権主張国 米国 (US)
- (31)優先権主張番号 08/482, 073
(32)優先日 1995年6月7日
(33)優先権主張国 米国 (US)
- (31)優先権主張番号 08/483, 958
(32)優先日 1995年6月7日
(33)優先権主張国 米国 (US)
- (31)優先権主張番号 08/484, 795
(32)優先日 1995年6月7日
(33)優先権主張国 米国 (US)
- (31)優先権主張番号 08/487, 801
(32)優先日 1995年6月7日
(33)優先権主張国 米国 (US)
- (81)指定国 EP (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, M C, NL, PT, SE), JP, KP, KR, US
- (72)発明者 パトカー, ニッティーン エー.